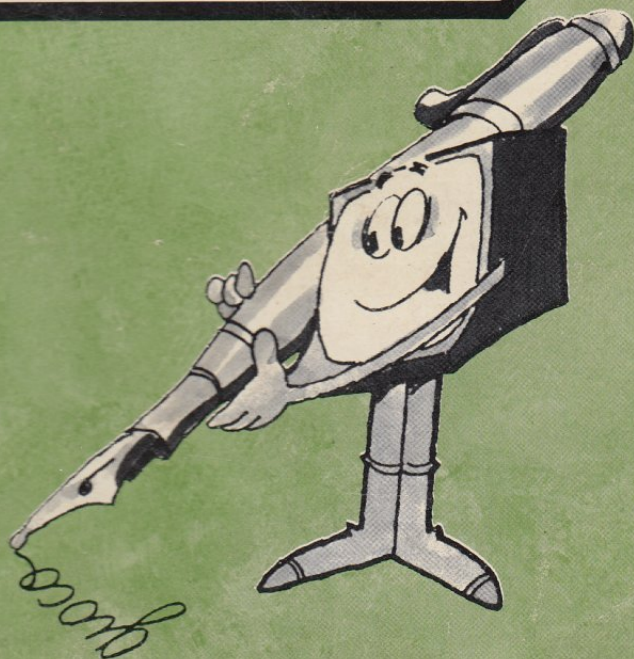
I QUADERNI JACKSON
di Personal Computer

Enrico Colombini

Scrivere un gioco di avventura sul personal computer

GRUPPO EDITORIALE JACKSON

Enrico Colombini

Scrivere un gioco di avventura sul personal computer

SOMMARIO

L'AUTORE

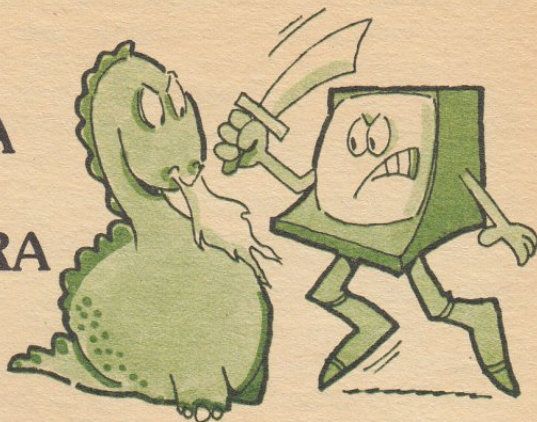
Enrico Colombini, 32 anni, autore di software e di pubblicazioni didattiche, vive e lavora a Brescia. Lasciati gli studi universitari ("poco spazio per la fantasia", ricorda), ha lavorato per vari anni come progettista elettronico indipendente (buona parte delle radio private, per dirne una, funzionano con circuiti di suo progetto). Nel 1980 è tra i fondatori della EC Elettronica (microprocessori ed automazione industriale), destinata ad una solida affermazione. Sempre inquieto ed in cerca dell'ultima frontiera, nel 1983 esce dall'azienda per occuparsi a tempo pieno di software (giochi intelligenti, soprattutto) e didattica (come questo libro). Fa parte dell'ABRS (Associazione Bresciana Ricerca Scientifica) e di Astrofisma (un'associazione scientifica specialmente attenta alla didattica extrascolastica). Nasconde sotto il suo caratteristico stile scanzonato una preparazione solida e aggiornata, dovuta ad un costante impegno professionale. Uno dei lati buoni di questo lavoro — dice — è che c'è sempre qualcosa da imparare. Altrimenti, avrebbe già cambiato mestiere un'altra volta.

PREFAZIONE

Questo libro vi insegna come scrivere un gioco di avventura con il vostro personal computer. Se non sapete cos'è un gioco di avventura, potete facilmente colmare una sì grave lacuna nella vostra formazione intellettuale con una semplice lettura del capitolo 1. Se siete già informati (o esperti avventurieri) potete saltare il capitolo in questione o leggerlo comunque per arricchire il vostro bagaglio culturale di ulteriori valigie, ehm, informazioni. Comunque, fate come vi pare: i lettori siete voi.
Sia detto di passata, mi propongo anche di insegnare un poco di Basic e di "stile" di programmazione. Ludendo docere (insegnare giocando), dicevano già molti secoli addietro: e così mi sento in buona compagnia nella mia avversione per ogni forma di insegnamento inutilmente noioso o "professorale", senza offesa (anzi con molto rispetto) per i professori veri, quelli che sanno attivare negli ascoltatori il più potente meccanismo di apprendimento: l'interesse.

Buon divertimento!

CAPITOLO 1

L'AVVENTURA DEI GIOCHI DI AVVENTURA

La prima avventura

In una data che non sono riuscito a scoprire con esattezza, ma in ogni caso precedente l'esistenza dei personal computer (più o meno nei primi anni '70), tali Crowther e Woods inventarono, o comunque realizzarono, un nuovo tipo di gioco, nel quale il calcolatore descrive un ambiente, o una situazione, con frasi come questa (traduco liberamente):

Sei in una splendida camera di una decina di metri d'altezza. I muri sono fiumi congelati di pietra arancione. Uno scosceso canyon ed un facile passaggio escono dai lati est ed ovest della camera.

C'è un allegro fringuello cinguettante.

Il giocatore, che si identifica nel personaggio che sta esplorando il luogo misterioso, può comunicare con il computer per mezzo di semplici frasi scritte alla tastiera, per esempio:

Cattura il fringuello

Ottenendo una risposta:

Il fringuello non aveva paura quando sei entrato, ma nell'avvicinarti si inquieta e non riesci a catturarlo.

Da questa risposta, il giocatore trae informazioni: ad esempio, che il fringuello ha paura di lui. Oppure, ha paura di qualche oggetto che egli possiede. Infatti, è anche possibile prendere e lasciare oggetti di vario tipo (lo scopo del gioco è trovare tutti i tesori ed uscire dalle caverne) e spostarsi da un luogo all'altro.
Il computer si comporta come se "capisse" le frasi del giocatore, dando

l'illusione di vivere realmente le situazioni descritte.
Crowther e Woods chiamarono il loro gioco semplicemente "Adventure", cioè "Avventura". Un nome azzeccato, che da allora viene impiegato per indicare tutti i giochi dello stesso genere.

Dai mini ai personal

Anche se il programma era scritto su un minicomputer, cioè una macchina relativamente piccola (per la precisione un DEC PDP-10), l'Adventure originale non era di certo destinata ad un folto pubblico: a quel tempo (come sembra lontano!) i computer erano macchine costosissime, accessibili soltanto a pochi specialisti in camice bianco e patrimonio di grandi industrie, università e centri di ricerca.
Tuttavia, il programma ebbe un grande successo tra i pochi fortunati che poterono giocarlo. Causò anche la disperazione di molti dirigenti, che vedevano i programmatori assorti in problemi ben diversi dal lavoro. Pare che ogni proibizione fosse inutile: l'unica terapia era attendere che il gioco venisse risolto, poi il lavoro poteva riprendere regolarmente.
Con l'introduzione del personal computer, vari programmatori si cimentarono nel compito di creare giochi di avventura che potessero funzionare sulle nuove macchine. La cosa non era facile: basti pensare che l'Adventure originale (scritto in linguaggio FORTRAN) occupava circa 300 Kbyte, quando sui personal c'erano sì e no 16 Kbyte disponibili. Uno dei precursori dei giochi di avventura sui personal fu Scott Adams, che già nel 1978 scrisse (in BASIC) un interprete che gli consentì di creare piccole ma ben congegnate avventure (se non sapete cosa sia un interprete, abbiate pazienza per qualche capitolo).
In seguito, l'evoluzione dei piccoli computer consentì di scrivere avventure sempre più complesse, incluse versioni dell'Adventure originale (per l'Apple II, ad esempio, ne esistono ben due: Apple Adventure e Microsoft Adventure, praticamente identiche).

Testo o grafica?

Il dialogo scritto non è il solo possibile mezzo di comunicazione tra giocatore e computer: un ambiente può essere disegnato invece che descritto. Esiste quindi anche un filone di avventure grafiche, ed esiste una disputa tra i sostenitori dei due tipi di gioco. Personalmente, trovo che entrambe le forme abbiano il loro fascino: se un'immagine colorata è più

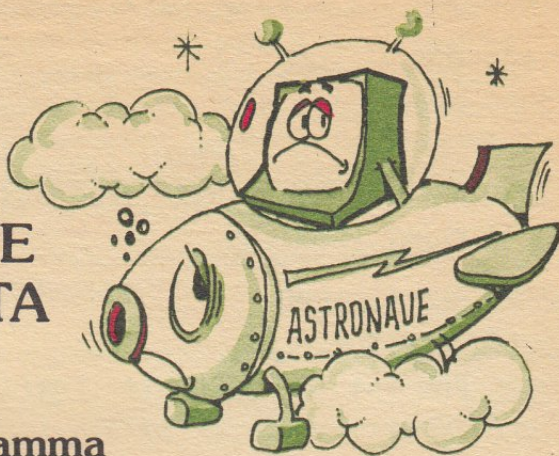
attraente di un testo scritto, è anche vero che il secondo stimola maggiormente la fantasia: nessun disegno sarà mai all'altezza di una scena immaginata sulla base di pochi spunti suggestivi. È la stessa differenza che passa tra un buon libro ed un bel film.
Io, in ogni caso, gioco volentieri sia avventure scritte che grafiche. Vi faccio notare che ho detto "un **buon** libro" ed "un **bel** film". Intendo dire che entrambi i tipi sono validi se ben fatti, ma c'è una vera inflazione di avventure decisamente scadenti (di entrambi i generi). Riprenderò più avanti questo argomento, per cercare di spiegare cosa può contribuire alla qualità di un'avventura.
In questo libro vi insegno come costruire avventure di tipo testo, ma gli stessi principi (come vedremo) sono perfettamente applicabili alla costruzione di avventure grafiche.

Qualche esempio

Se già non li conoscete, vi suggerisco qualche titolo tra le migliori avventure in circolazione. Se esiste per il vostro calcolatore, procuratevi senz'altro l'Adventure originale: ne vale la pena. Ovviamente è in inglese (o meglio, in americano), come d'altronde la stragrande maggioranza delle avventure. Non lasciatevi scoraggiare: è un ottimo stimolo per migliorare la conoscenza della lingua. Se volete saperlo, a scuola ho studiato soltanto francese: una lingua interessante, ma del tutto inadatta alle necessità di qualunque professione scientifica. Ho iniziato a conoscere un buon numero di vocaboli inglesi proprio grazie ad "Apple Adventure". Non ho mai sfogliato tanto un dizionario in vita mia, ma ne valeva la pena. Le avventure di Scott Adams sono invece scritte in inglese molto sintetico, per occupare il minimo di memoria. Si trovano perfino in cartuccia per VIC-20. Interessanti, tra le altre, "Pirate's adventure" (la ricerca di un tesoro) e "Mission impossible" (un reattore nucleare impazzito), che recentemente è stata ribattezzata con altro nome per non confonderla con un omonimo videogame.

La più complessa ed affascinante avventura di tipo testo è senza dubbio "Zork", divisa in tre distinte avventure. Si svolge in un immaginario impero sotterraneo (the Great Underground Empire), ed è scritta in un inglese molto ricercato, quasi letterario. È molto bella, ma anche molto difficile. Tra le avventure grafiche, vi consiglio "Ulysses and the golden fleece" (Ulisse e il vello d'oro) e "Dark Crystal", entrambe pubblicate dalla Online Systems. Se avete molta pazienza, c'è anche la colossale "Time Zone" (12 dischi nella versione Apple II!), anche se un po' meno appas-

sionante per l'eccessiva dispersione dei problemi da risolvere.
Un'avventura che non vi consiglio è la pubblicizzatissima The Hobbit, che dietro un'apparente intelligenza nasconde una notevole stupidità, parecchi errori di programmazione e la totale assenza del fascino del libro originale di Tolkien. Potete consolarvi con i disegni, che sono ben fatti.
Se volete qualcosa in italiano, vi consiglio naturalmente "Avventura nel castello", opera del sottoscritto insieme con Chiara Tovena (grazie, basta applausi) e "Il mistero della piramide" di Enrico Ragaini. Non è detto che le avventure italiane siano peggio di quelle USA. Giocare per credere.

L'ASTRONAVE CONDANNATA

Introdurre il programma

Non è mia intenzione annoiarvi con astrusi discorsi teorici. Invece, vi propongo subito un'avventura da giocare. Nei capitoli successivi vi mostrerò come, un passo alla volta, sia nata questa avventura. Imparerete così a costruire le vostre avventure, sfruttando una parte (la più complessa) del programma che vi propongo, ed aggiungendovi i frutti della vostra fantasia. Infine, potrete vedere come funziona il programma e come modificarlo per le vostre esigenze particolari. Ci sarà anche una certa dose di teoria, ma diluita e (spero) facilmente digeribile. Questo libro richiede una conoscenza (almeno superficiale) del Basic. Se non l'avete, o se volete migliorarla, leggetevi ABC personal computer o i Quaderni Jackson n. 3, 5 e 9.
Il listato dell'avventura è riportato al termine di questo capitolo. Se non avete voglia di battere sulla tastiera le oltre 350 linee che costituiscono il programma, potete trovarlo già pronto in "JACKSON SOFT AVVENTURE 1", anche leggermente modificato.

Altrimenti, armatevi di santa pazienza e mettetevi alla tastiera. Il listato di seguito pubblicato è nella versione per Apple II, ma per le altre macchine bastano poche varianti (che elenco). Unica eccezione: non fornisco la versione per il Sinclair Spectrum, che usa un Basic del tutto fuori standard ed alquanto limitato. Se avete lo Spectrum, non vi resta che procurarvi il numero citato di JACKSON SOFT AVVENTURE, la cui cassetta allegata contiene una versione del gioco adeguatamente modificata.

Per **Apple II** (tutte le versioni): nessuna variazione.

Per **Commodore 64** o **VIC-20** espanso (almeno 16K), togliere il punto

di domanda nelle INPUT alla linea 800 e 1330, e sostituire le linee 1990-2090 con quelle riportate in Figura 1.

```
1990 REM 5: SAVE
2000 OPEN 1,1,1,F$
2010 FOR I=1 TO FO:PRINT#1,LO%(I):NEXT
2020 PRINT#1,LU:PRINT#1,T1:PRINT#1,V1:PRINT#1,V2
2030 CLOSE 1:RETURN

2050 REM 6: LOAD
2060 OPEN 1,1,0,F$
2070 FOR I=1 TO FO:INPUT#1,LO%(I):NEXT
2080 INPUT#1,LU,T1,V1,V2
2090 CLOSE 1:RETURN
```

Fig. 1 — Routine di Save e Load per Commodore 64 e Vic 20 espanso.

Per **MSX, IBM, Olivetti, Macintosh** con Microsoft Basic, togliere il punto di domanda nelle INPUT alla linea 800 e 1330, sostituire la linea 1130 con:

1130 A$=INPUT$(1)

e sostituire le linee 1990-2090 con quelle riportate in Figura 2. Ci

```
1990 REM 5: SAVE
2000 OPEN F$ FOR OUTPUT AS #1
2010 FOR I=1 TO FO:PRINT#1,LO%(I):NEXT
2020 PRINT#1,LU:PRINT#1,T1:PRINT#1,V1:PRINT#1,V2
2030 CLOSE#1:RETURN

2050 REM 6: LOAD
2060 OPEN F$ FOR INPUT AS #1
2070 FOR I=1 TO FO:INPUT#1,LO%(I):NEXT
2080 INPUT#1,LU,T1,V1,V2
2090 CLOSE#1:RETURN
```

Fig. 2 — Routine di Save e Load per versioni recenti di Microsoft Basic.

possono essere problemi con le lettere minuscole. Se il vostro computer non le possiede o volete evitare qualsiasi problema, scrivete semplicemente tutto in maiuscolo. Se invece avete il Commodore 64, rischiate di trovarvi invischiati in una mostruosità di progetto che la Commodore si trascina da anni: lavorando in modo "solo maiuscole" va tutto bene, ma passando in modo "maiuscole/minuscole" i codici dei caratteri vengono scambiati su video (e lasciati inalterati su tastiera). Se volete usare il C-64 in modo "maiuscole/minuscole", dovete:

- Mettere in modo "maiuscole/minuscole" con SHIFT/C=
- Scrivere il programma in minuscolo.
- Scrivere i REM in minuscolo.
- Scrivere quello che sta tra virgolette come appare nel listato, con le seguenti eccezioni:
- Linee 1340,1350: "S" ed "N" vanno minuscole.
- Linea 830: "RE" va minuscolo.
- Linee 3990-4090: tutto minuscolo.
- Linee 4090, 4230, 4360, 4510: "FD", "FM", "FA", "FO" minuscoli. Lo stesso nelle linee 1040, 1050, 1060, 1070.

Nell'inserire il programma, fate attenzione alle solite cose: a non confondere la lettera O con la cifra zero (che è di solito barrata proprio per distinguerla), a non tralasciare gli spazi nelle stringhe (quelli tra virgolette), a non dimenticare due punti, virgole o punti e virgola. Se volete, potete risparmiarvi le linee di commento (quelle che cominciano con REM). Quando avete finito, fatevi aiutare da qualcuno per verificare bene il tutto.

Giocare l'avventura

Quando avete il programma in macchina (ricordatevi di salvarne una copia se l'avete battuto a mano), potete dare RUN e giocare. Non vi conviene andare avanti nella lettura del libro prima di aver giocato, perché trovereste le soluzioni dell'avventura (rovinandovi il divertimento) ed avreste qualche difficoltà a comprendere i continui riferimenti all'avventura contenuta nel programma.
Per chi non fosse esperto di avventure, ecco le istruzioni:

Siete divisi in una doppia personalità, che potremmo chiamare il braccio e la mente. Dovete dare ordini al voi stesso che vive dentro il calcolatore e che si trova a dover affrontare il problema di un'astronave condannata a sicura fine. Potete farlo con frasi del tipo:

GUARDA IL COMODINO

(ricordatevi di darvi sempre del tu) oppure indicando la direzione in cui volete muovervi, ad esempio:

NORD

o, più semplicemente:

N

Le direzioni possibili sono NORD, SUD, EST, OVEST, ALTO, BASSO, abbreviabili rispettivamente in N, S, E, O, A, B. Provatele in tutti i luoghi e disegnate una buona piantina.
Un'azione fondamentale è GUARDA, che fornisce informazioni tipo:

GUARDA IL CARTELLO

Altre azioni essenziali sono PRENDI e LASCIA, per manipolare gli oggetti:

PRENDI LA CHIAVE

Per ottenere l'elenco degli oggetti posseduti, scrivete COSA o INVENTARIO.
Se volete registrare la situazione su disco (o cassetta) per continuare in seguito, scrivete:

SAVE

Questo comando è anche utile prima di compiere un'azione pericolosa, per non dover ricominciare dall'inizio. Se vi capita qualche sgradevole incidente, basta poi scrivere:

LOAD

per riprendere la situazione esattamente com'era al punto dell'ultimo SAVE effettuato in precedenza.
Naturalmente i vocaboli citati non esauriscono tutte le possibilità. Se proprio non trovate l'azione giusta da fare su un'oggetto, potete dare una sbirciatina alle linee dalla 3990 alla 4090, che contengono le parole ammesse.
L'avventura è molto semplice (a parte forse l'azione conclusiva, che richiede un po' di ragionamento), ma abbastanza curata e "realistica".
Spero vi divertiate a giocarla, e spero anche che la risolviate senza troppi

problemi. In ogni caso, non disperate: nei prossimi capitoli è descritta in dettaglio. Certo che sarebbe meglio riuscire a risolverla da soli...
Ecco il programma, in versione per Apple II (per gli altri computer, vedi sopra):

```
100 REM ##  AVVENTURA: L'ASTRONAVE CONDANNATA   ##
11Ø REM ## di Enrico Colombini e Chiara Tovena  ##
12Ø REM ## (C) 1985 DINOSOFT E JACKSON EDITRICE ##
13Ø REM
14Ø GOTO 71Ø:REM MAIN
15Ø REM
16Ø REM - CERCA P$ IN DIZIONARIO, C=CODICE (Ø SE ASSENTE) -
17Ø REM
18Ø I=1:F=FD
19Ø A=INT((I+F)/2):A$=DZ$(A):IF P$=A$ THEN C=DZ(A):GOTO 24Ø
2ØØ IF P$>A$ THEN I=A+1
21Ø IF P$<A$ THEN F=A-1
22Ø IF I<=F THEN 19Ø
23Ø C=Ø
24Ø RETURN
25Ø REM
26Ø REM - ESTRAE P$ DA IN$[IN], TROVA CODICE C, SALTA ART. -
27Ø REM
28Ø C=Ø
29Ø C$=MID$(IN$,IN,1):IF C$=" " OR C$="'" THEN IN=IN+1:
    GOTO 29Ø
3ØØ IF IN>LI THEN P$="":GOTO 36Ø
31Ø A=IN:REM INIZIO
32Ø C$=MID$(IN$,IN,1)
33Ø IF C$<>" " AND C$<>"'" AND IN<=LI THEN IN=IN+1:GOTO 32Ø
34Ø P$=MID$(IN$,A,IN-A):GOSUB 18Ø:REM CERCA
35Ø IF C=7 THEN 28Ø:REM ARTICOLO
36Ø RETURN
37Ø REM
38Ø REM - CERCA AZIONE A, A=AZIONE (Ø SE NON TROVATA) -
39Ø REM
4ØØ I=1:F=FA:N=A
41Ø A=INT((I+F)/2):M=CA(A):IF N=M THEN A=AZ%(A):GOTO 46Ø
42Ø IF N>M THEN I=A+1
43Ø IF N<M THEN F=A-1
44Ø IF I<=F THEN 41Ø
45Ø A=Ø
46Ø RETURN
47Ø REM
48Ø REM - ESEGUE AZIONE A, A=Ø SE NON TROVATA -
49Ø REM
5ØØ GOSUB 4ØØ:IF A=Ø THEN 55Ø
51Ø IF C2=Ø OR A>Ø THEN 53Ø:REM SOLO VERBO O NO TEST
52Ø A=-A:IF OG=Ø THEN PRINT "- Qui non c'e'.":GOTO 54Ø
53Ø GOSUB 171Ø:REM ESEGUE
54Ø A=1
55Ø RETURN
56Ø REM
57Ø REM - ELENCA OGGETTI IN LUOGO L, CON PREFISSO P$ -
58Ø REM
```

```
590 FOR I=1 TO FO
600 IF ABS(LO%(I))=L THEN PRINT P$;OG$(I);"."
610 NEXT:RETURN
620 REM
630 REM - OG=INDICE OGGETTO C2, Ø SE NON PRES. O TRASP. -
640 REM
650 OG=Ø:FOR I=1 TO FO
660 IF CO%(I)=C2 THEN IF ABS(LO%(I))=LU OR LO%(I)=Ø THEN
    OG=I:I=FO
670 NEXT:RETURN
680 REM
690 REM - MAIN (PARSER) -
700 REM
710 GOSUB 970:REM INIT
720 GOSUB 1540:REM INTRODUZIONE
730 GOSUB 1120:REM <SP>
740 REM
750 REM CICLO DI GIOCO:
760 PRINT:PRINT "Sei ";DE$(LU);".":REM DESCRIZIONE
770 L=LU:P$="Vedo ":GOSUB 590:REM OGGETTI
780 GOSUB 1400:REM TEMPO
790 PRINT:PRINT
800 IN$="":INPUT "Cosa devo fare ? ";IN$:IF IN$="" THEN 800
810 PRINT:LI=LEN(IN$):IN=1:GOSUB 280:P1$=P$:C1=C
820 IF P1$="" THEN PRINT "- Beh ?":GOTO 760
830 IF RIGHT$(P1$,2)="RE" THEN PRINT
    "- Dammi del tu, per favore.":GOTO 760
840 IF C1=Ø AND P1$<>"" THEN PRINT
    "- Non conosco il verbo '"P1$"'.":GOTO 760
850 GOSUB 280:P2$=P$:C2=C
860 IF C2=Ø AND P2$<>"" THEN PRINT
    "- Non conosco la parola '"P2$"'.":GOTO 760
870 IF C2<>Ø THEN GOSUB 650:REM OG=INDICE
880 N1=LU*10000:N2=C1*100
890 A=N1+N2+C2:GOSUB 500:IF A GOTO 760:REM VERBO+NOME IN LU
900 IF C2<>Ø THEN A=N1+N2+99:GOSUB 500:IF A GOTO 760:REM
    VERBO+X IN LU
910 A=N2+C2:GOSUB 500:IF A GOTO 760:REM VERBO+NOME GENERICO
920 IF C2<>Ø THEN A=N2+99:GOSUB 500:IF A GOTO 760:REM
    VERBO+X GENERICO
930 PRINT "- Non capisco.":GOTO 760
940 REM
950 REM - INIT -
960 REM
970 DIM DZ$(100),DZ(100):REM DIZIONARIO
980 DIM DE$(30),DI$(30):REM MAPPA
990 DIM CA(150),AZ%(150):REM AZIONI
1000 DIM OG$(50),CO%(50),LO%(50):REM OGGETTI
1010 F$="ASTRO":REM FILE SITUAZIONE
1020 REM FD,FM,FA,FO=Ø
1030 PRINT:PRINT "Un attimo di pazienza...";
1040 READ A$:IF A$<>"FD" THEN FD=FD+1:DZ$(FD)=A$:
     READ DZ(FD):GOTO 1040
1050 READ A$:IF A$<>"FM" THEN FM=FM+1:DE$(FM)=A$:
     READ DI$(FM):GOTO 1050
1060 READ A$:IF A$<>"FA" THEN FA=FA+1:CA(FA)=VAL(A$):
     READ AZ%(FA):GOTO 1060
```

```
1070 READ A$:IF A$<>"FO" THEN FO=FO+1:OG$(FO)=A$:
     READ CO%(FO),LO%(FO):GOTO 1070
1080 PRINT:PRINT:RETURN
1090 REM
1100 REM - <SP> PER CONTINUARE -
1110 REM
1120 PRINT:PRINT "- Premi <spazio> per continuare -";
1130 GET A$:IF A$<>" " THEN 1130
1140 PRINT:PRINT:RETURN
1150 REM
1160 REM ### FINE INTERPRETE, INIZIO AVVENTURA ###
1170 REM
1180 REM - MORTO -
1190 REM
1200 GOSUB 1120:REM <SP>
1210 PRINT "### IL GAZZETTINO DELLA GALASSIA ###
1220 PRINT:PRINT "- Tragedia al largo di Vega -":PRINT
1230 PRINT "L'astronave 'Neutronia', in servizio"
1240 PRINT "passeggeri con 250 persone a bordo,"
1250 PRINT "e' stata distrutta da una violenta"
1260 PRINT "esplosione, causata probabilmente"
1270 PRINT "da imperizia del comandante (un"
1280 PRINT "(novellino, stando a indiscrezioni"
1290 PRINT "raccolte dal nostro corrispondente)."
1300 PRINT "Il responsabile del peggior disastro"
1310 PRINT "della storia galattica sara' di certo"
1320 PRINT "degradato alla memoria.":PRINT
1330 INPUT "Vuoi giocare ancora ? ";A$
1340 A$=LEFT$(A$,1):IF A$<>"S" AND A$<>"N" THEN 1330
1350 IF A$="S" THEN RUN:REM RIPARTE
1360 PRINT:PRINT:PRINT "Ciao!":PRINT:END
1370 REM
1380 REM - TEMPO -
1390 REM
1400 T1=T1-1
1410 IF T1<=10 THEN PRINT "Sento un segnale di allarme."
1420 IF T1<=5 THEN PRINT "La temperatura e' insopportabile."
1430 IF T1<=2 THEN PRINT "La nave e' scossa da vibrazioni."
1440 IF T1>0 THEN RETURN
1450 PRINT:PRINT "*** Troppo tardi! ***":PRINT
1460 PRINT "Il reattore e' fuori controllo."
1470 PRINT:PRINT "L'astronave e' disintegrata"
1480 PRINT "in minuscoli frammenti."
1490 PRINT "E' incredibile il silenzio"
1500 PRINT "delle esplosioni nel vuoto.":GOTO 1200
1510 REM
1520 REM - INTRODUZIONE -
1530 REM
1540 PRINT:PRINT:PRINT
1550 PRINT "# L'astronave condannata:un'avventura #"
1560 PRINT "# di Enrico Colombini e Chiara Tovena #"
1570 PRINT:PRINT "(c) Dinosoft e Jackson editrice 1985":
     PRINT
1580 PRINT "Sdraiato nella sabbia, ti godi il"
1590 PRINT "dolce calore del sole tropicale..."
1600 PRINT:PRINT "Ora il sole picchia piu' forte,"
```

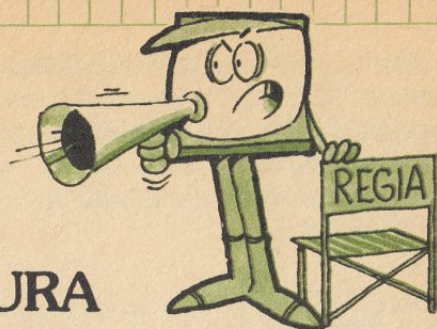
```
1610 PRINT "sei in pieno deserto e non c'e'"
1620 PRINT "traccia di oasi..."
1630 PRINT:PRINT "Ti svegli di soprassalto nella tua"
1640 PRINT "cabina di comandante del 'Neutronia'."
1650 PRINT "Fa molto caldo. Troppo caldo. Ci"
1660 PRINT "dev'essere qualcosa che non funziona."
1670 T1=100:LU=6:V1=0:V2=0:PRINT:RETURN
1680 REM
1690 REM - ESEGUE AZIONE A -
1700 REM
1710 ON A GOTO 1780,1830,1920,1970,2000,2060,2120,2150,
     2180,2210
1720 ON A-10 GOTO 2260,2300,2380,2410,2460,2580,2640,2700,
     2740,2800
1730 ON A-20 GOTO 2860,2910,2960,3010,3040,3080,3190,3320,
     3370,3400
1740 ON A-30 GOTO 3440,3520,3570,3640,3800,3860,3900,3940
1750 PRINT "AZIONE "A:RETURN:REM  TEST
1760 REM
1770 REM 1:DIREZIONI
1780 A=VAL( MID$(DI$(LU),2*C1-1,2))
1790 IF A=0 THEN PRINT "- Di li' non puoi andare.":RETURN
1800 LU=A:RETURN
1810 REM
1820 REM 2:PRENDI
1830 IF LO%(OG)=0 THEN PRINT "- Gia' fatto.":RETURN
1840 IF LO%(OG)<0 THEN PRINT "- Non e' possibile.":RETURN
1850 IF OG=4 AND LO%(22)=0 THEN PRINT
     "Togli prima il camice.":RETURN
1860 IF OG=22 AND LO%(4)=0 THEN PRINT
     "Togli prima la tuta.":RETURN
1870 LO%(OG)=0
1880 IF OG=4 OR OG=9 OR OG=11 THEN PRINT
     "Ora l'hai addosso.":RETURN
1890 PRINT "Fatto.":RETURN
1900 REM
1910 REM 3: LASCIA
1920 IF OG=0 OR LO%(OG)<>0 THEN PRINT
     "- Non ce l'hai.":RETURN
1930 IF LU<9 THEN LO%(OG)=LU:PRINT "Fatto.":RETURN
1940 LO%(OG)=-99:PRINT "Si e' perso nello spazio.":RETURN
1950 REM
1960 REM 4: GUARDA
1970 PRINT "Non noto nulla di particolare.":RETURN
1980 REM
1990 REM 5: SAVE
2000 D$=CHR$(4):PRINT D$"OPEN"F$:PRINT D$"WRITE"F$
2010 FOR I=1 TO FO:PRINT LO%(I):NEXT
2020 PRINT LU:PRINT T1:PRINT V1:PRINT V2
2030 PRINT D$"CLOSE":RETURN
2040 REM
2050 REM 6: LOAD
2060 D$=CHR$(4):PRINT D$"OPEN"F$:PRINT D$"READ"F$
2070 FOR I=1 TO FO:INPUT LO%(I):NEXT
2080 INPUT LU,T1,V1,V2
2090 PRINT D$"CLOSE":RETURN
```

```
2100 REM
2110 REM 7: COSA
2120 PRINT "Possiedi:":L=Ø:P$="- ":GOTO 59Ø
2130 REM
2140 REM 8:
2150 RETURN
2160 REM
2170 REM 9:
2180 RETURN
2190 REM
2200 REM 1Ø:
2210 IF LO%(OG)<>Ø THEN PRINT "Non ce l'hai.":RETURN
2220 IF NOT(LU>=9 OR(LU=7 AND V2=1)) GOTO 192Ø:REM LASCIA
2230 PRINT "L'aria! L'aria! Aaaagh!!!":GOTO 12ØØ
2240 REM
2250 REM 11:
2260 PRINT "E' la tua tuta per attivita'"
2270 PRINT "extraveicolare.":RETURN
2280 REM
2290 REM 12:
2300 PRINT "E' privo di conoscenza ed ha"
2310 PRINT "un'ammaccatura nel casco."
2320 PRINT "Probabilmente e' stato colpito"
2330 PRINT "da un piccolo meteorite mentre"
2340 PRINT "riparava l'antenna. Per"
2350 PRINT "fortuna, e' ancora vivo.":RETURN
2360 REM
2370 REM 13:
2380 PRINT "Non e' meglio leggerlo ?":RETURN
2390 REM
2400 REM 14:
2410 PRINT "E' piuttosto pesante."
2420 PRINT "Probabilmente, e'"
2430 PRINT "trattato al piombo.":RETURN
2440 REM
2450 REM 15:
2460 IF LO%(OG)=LU THEN PRINT
     "Prendilo in mano, prima.":RETURN
2470 PRINT "- MANUALE DI ISTRUZIONI DEL -"
2480 PRINT "-    REATTORE POSITRONICO    -"
2490 PRINT "-       Mod. YTREWQ 8421      -":PRINT
2500 PRINT "- Per attivare il reattore,"
2510 PRINT "tirare la leva e poi premere"
2520 PRINT "in sequenza i pulsanti verde,"
2530 PRINT "giallo e rosso."
2540 PRINT "- Per disattivare il reattore..."
2550 PRINT:PRINT "Dannazione! La pagina e' strappata.":
     RETURN
2560 REM
2570 REM 16:
2580 PRINT "- TEMPERATURA REATTORE -":PRINT
2590 PRINT "Segna "84Ø-T1*4" gradi e sta"
2600 PRINT "salendo velocemente. C'e'"
2610 PRINT "un segno rosso a 8ØØ gradi.":RETURN
2620 REM
2630 REM 17:
```

```
2640 PRINT "- S.O.S. GALATTICO -":PRINT
2650 PRINT "Premere il pulsante solo"
2660 PRINT "in caso di emergenza."
2670 PRINT "Ogni abuso verra' punito.":RETURN
2680 REM
2690 REM 18:
2700 PRINT "Una scritta lampeggia brevemente:":PRINT
2710 PRINT "- ANTENNA ESTERNA DIFETTOSA -":RETURN
2720 REM
2730 REM 19:
2740 PRINT "E' alla base di una scaletta"
2750 PRINT "e dice:":PRINT
2760 PRINT "INGRESSO RISERVATO AL"
2770 PRINT "PERSONALE DI BORDO.":RETURN
2780 REM
2790 REM 20:
2800 LU=1:IF LO%(20)<>9 THEN RETURN
2810 PRINT "Se solo ci fosse qui il secondo"
2820 PRINT "pilota, il solo che se ne intende"
2830 PRINT "di problemi tecnici!":RETURN
2840 REM
2850 REM 21:
2860 PRINT "Meglio non svegliare i"
2870 PRINT "passeggeri, potrebbero"
2880 PRINT "farsi prendere dal panico.":RETURN
2890 REM
2900 REM 22:
2910 PRINT "E' posto ad ovest e dice:":PRINT
2920 PRINT "- ATTENZIONE: -":PRINT
2930 PRINT "STANZA DEPRESSURIZZATA":RETURN
2940 REM
2950 REM 23:
2960 IF LO%(24)<>-99 THEN PRINT "Gia' fatto.":RETURN
2970 IF LO%(23)<>0 THEN PRINT "E' chiuso a chiave.":RETURN
2980 PRINT "Fatto.":LO%(24)=LU:LO%(22)=LU:RETURN
2990 REM
3000 REM 24:
3010 PRINT "E' vuoto.":RETURN
3020 REM
3030 REM 25:
3040 PRINT "Dimmi: 'Premi il rosso'"
3050 PRINT "o 'Premi il verde'.":RETURN
3060 REM
3070 REM 26:
3080 IF V2=1 THEN PRINT "Click.":RETURN
3090 PRINT "La parete ad est si chiude."
3100 PRINT "La parete ad ovest si apre"
3110 PRINT "verso lo spazio esterno."
3120 PRINT "L'aria esce sibilando."
3130 IF LO%(4)<>0 OR LO%(11)<>0 THEN PRINT:PRINT
     "Aaaagh!":GOTO 1200
3140 FOR I=1 TO FO
3150 IF LO%(I)=LU THEN PRINT
     OG$(I)" si perde nello spazio.":LO%(I)=-99
3160 NEXT:V2=1:RETURN
3170 REM
```

```
3180 REM 27:
3190 IF V2=Ø THEN PRINT "Click.":RETURN
3200 PRINT "La parete ad ovest si chiude."
3210 PRINT "La parete ad est si apre"
3220 PRINT "verso il corridoio."
3230 PRINT "L'aria rientra sibilando.":V2=Ø
3240 IF LO%(2Ø)<>Ø OR LO%(23)<>-99 THEN RETURN
3250 PRINT:PRINT "Il secondo pilota rinviene, si rende"
3260 PRINT "subito conto della situazione e dice:":PRINT
3270 PRINT "- Presto, ferma il reattore!"
3280 PRINT "Ecco la chiave del mio..."
3290 PRINT "Poi perde nuovamente i sensi.":
     LO%(23)=LU:RETURN
3300 REM
3310 REM 28:
3320 PRINT "- L'EMERGENZA E' IN AGGUATO! -"
3330 PRINT "-     Porta sempre con te      -"
3340 PRINT "-  il manuale del reattore.  -":RETURN
3350 REM
3360 REM 29:
3370 PRINT "Il tecnico e' il secondo pilota.":RETURN
3380 REM
3390 REM 3Ø:
3400 PRINT "Dimmi: 'Premi il rosso', 'Premi"
3410 PRINT "il verde' o 'Premi il giallo'.":RETURN
3420 REM
3430 REM 31:
3440 PRINT "Click.":IF V1<>Ø GOTO 383Ø
3450 PRINT "Una tubatura perde leggermente"
3460 PRINT "da un raccordo (probabilmente per"
3470 PRINT "sovrapressione). Le gocce cadono sul"
3480 PRINT "quadro di controllo, vicino a te."
3490 V1=1:RETURN
3500 REM
3510 REM 32:
3520 PRINT "Click.":IF V1<>1 GOTO 383Ø
3530 V1=2:IF LO%(22)<>Ø THEN PRINT
     "Non ti senti troppo bene."
3540 RETURN
3550 REM
3560 REM 33:
3570 PRINT "Click.":IF V1<>2 GOTO 383Ø
3580 V1=3:IF LO%(22)=Ø THEN RETURN
3590 PRINT "Temo che tu abbia assorbito troppe"
3600 PRINT "radiazioni. Ora stai decisamente"
3610 PRINT "male. Perdi conoscenza...":GOTO 12ØØ
3620 REM
3630 REM 34:
3640 PRINT "Clank.":IF V1<>3 GOTO 383Ø
3650 GOSUB 112Ø:REM <SP>
3660 PRINT "### IL GAZZETTINO DELLA GALASSIA ###
3670 PRINT:PRINT "- Comandante salva astronave -":PRINT
3680 PRINT "L'astronave 'Neutronia', in servizio"
3690 PRINT "passeggeri con 25Ø persone a bordo,"
3700 PRINT "e' stata salvata da sicura distruzione"
3710 PRINT "grazie al coraggio e al sangue freddo"
```

```
372Ø PRINT "del comandante, che e' riuscito a"
373Ø PRINT "disattivare il reattore impazzito."
374Ø PRINT "Il suo nome verra' ricordato per"
375Ø PRINT "sempre tra gli eroi della nostra"
376Ø PRINT "flotta galattica.":PRINT
377Ø PRINT:PRINT "- Congratulazioni! -":PRINT:END
378Ø REM
379Ø REM 35:
38ØØ PRINT "Clunk.":GOTO 383Ø
381Ø REM
382Ø REM COMUNE:
383Ø V1=Ø:T1=INT(T1/2):RETURN
384Ø REM
385Ø REM 36:
386Ø PRINT "Sembra danneggiata, forse"
387Ø PRINT "da un grosso meteorite.":RETURN
388Ø REM
389Ø REM 37:
39ØØ IF V2=Ø THEN LU=4:RETURN
391Ø GOTO 178Ø:REM DIREZIONI
392Ø REM
393Ø REM 38:
394Ø IF V2=1 THEN LU=11:RETURN
395Ø GOTO 178Ø:REM DIREZIONI
396Ø REM
397Ø REM - DIZIONARIO:
398Ø REM
399Ø DATA "A",5,"AGGIUSTA",27,"ALTO",5,"ANTENNA",69,
     "APRI",22
4ØØØ DATA "ARMADIETTO",67,"B",6,"BASSO",6,"BOTTONE",61,
     "CAMICE",52
4Ø1Ø DATA "CARTELLO",6Ø,"CASCO",5Ø,"CHIAVE",54,"COSA",13,
     "E",3,"EST",3
4Ø2Ø DATA "ETICHETTA",7Ø,"GIALLO",63,"GUARDA",1Ø,"IL",7
4Ø3Ø DATA "INDICATORE",66,"INDOSSA",2Ø,"INVENTARIO",13,
     "L",7,"LA",7
4Ø4Ø DATA "LASCIA",9,"LEGGI",25,"LETTO",68,"LEVA",65,
     "LO",7,"LOAD",12
4Ø5Ø DATA "MANUALE",55,"METTI",2Ø,"N",1,"NORD",1,"O",4,
     "OVEST",4
4Ø6Ø DATA "PREMI",26,"PRENDI",8,"PULSANTE",61,"RIPARA",27,
     "ROSSO",64
4Ø7Ø DATA "S",2,"SALI",5,"SAVE",11,"SCENDI",6,
     "SCHIACCIA",26
4Ø8Ø DATA "SECONDO",53,"SPINGI",24,"SUD",2,"TIRA",23,
     "TOGLI",21
4Ø9Ø DATA "TUTA",51,"VERDE",62,"W",4,"FD"
41ØØ REM
411Ø REM - MAPPA:
412Ø REM
413Ø DATA "nella cabina di pilotaggio","ØØØØØØØØØØ2"
414Ø DATA "ad un'estremita' del corridoio","ØØØ3ØØØ5Ø1ØØ"
415Ø DATA "nel corridoio","Ø2Ø4ØØØ6ØØØØ"
416Ø DATA "ad un'estremita' del corridoio","Ø3ØØØØ7ØØØ8"
417Ø DATA "nella cabina del secondo pilota","ØØØØ2ØØØØØØ"
418Ø DATA "nella tua cabina","ØØØØ3ØØØØØØ"
```

```
4190 DATA "nel compartimento stagno","000000000000"
4200 DATA "nella sala controllo del reattore","00000000400"
4210 DATA "all'esterno, a prùa dell'astronave",
     "00100000000"
4220 DATA "all'esterno dell'astronave","09110000000"
4230 DATA "all'esterno, a poppa dell'astronave",
     "10000700000","FM"
4240 REM
4250 REM - AZIONI:
4260 REM
4270 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2
4280 DATA 950,-10,951,-10,999,3,1051,-11,1052,-14,1053,-12
4290 DATA 1055,-13,1060,-13,1099,-4,1100,5,1200,6,1300,7
4300 DATA 2050,-2,2051,-2,2052,-2,2150,-10,2151,-10,
     2152,-3,2255,-15
4310 DATA 2555,-15,2769,-29,6550,-10,6551,-10,6552,-3
4320 DATA 11066,16,12570,17,12661,18,20300,21,20500,20,
     22560,19
4330 DATA 30300,21,40300,21,42560,22,52267,23,62267,24
4340 DATA 70300,37,70400,38
4350 DATA 72661,25,72662,27,72664,26,81066,16,82365,35,
     82465,34
4360 DATA 82560,28,82661,30,82662,33,82663,32,82664,31,
     91069,36,"FA"
4370 REM
4380 REM - OGGETTI:
4390 REM
4400 DATA "un indicatore",66,-1,"un pulsante",61,-1
4410 DATA "un'etichetta",70,-1,"una tuta",51,1
4420 DATA "un cartello bianco",60,-2,
     "un cartello giallo",60,-4
4430 DATA "un letto",68,-5,"un armadietto",67,-5
4440 DATA "un letto",68,-6,"un armadietto",67,-6
4450 DATA "un casco",50,6,"un pulsante rosso",61,-7
4460 DATA "un pulsante verde",61,-7,"un indicatore",66,-8
4470 DATA "una leva",65,-8,"un pulsante rosso",61,-8
4480 DATA "un pulsante verde",61,-8,
     "un cartello rosso",60,-8
4490 DATA "un pulsante giallo",61,-8,
     "il secondo pilota",53,9
4500 DATA "un'antenna parabolica",69,-9,"un camice",52,-99
4510 DATA "una chiave",54,-99,"un manuale",55,-99,"FO"
```

CAPITOLO 3

LA SCENEGGIATURA

Attenzione: questo capitolo contiene le soluzioni dell'avventura. Se lo leggete prima di averla giocata, vi perdete tutto il divertimento.

L'idea

Realizzare un'avventura richiede fantasia e tecnica, esattamente come scrivere un racconto, dipingere un quadro, o qualunque altra forma espressiva. Entrambe le componenti sono fondamentali per una buona riuscita del lavoro.
È chiaro che io posso fornirvi soltanto gli strumenti tecnici, non certo la fantasia o la creatività. Posso però esservi d'aiuto nell'organizzare le idee nella fase iniziale di creazione del gioco, mostrandovi come è nata "L'astronave condannata".
Per prima cosa occorre un'idea, un'ambientazione: dove si svolge l'avventura? Qual è il soggetto della storia?
Era un sabato mattina d'inverno, con 85 centimetri di neve fuori dalla porta di casa (ma non divaghiamo, questa è un'altra avventura). Era un sabato mattina, dicevo, e stavo discutendo con Chiara su quale potesse essere il soggetto della piccola avventura dimostrativa da includere in questo libro. La prima idea riguardava la ricerca di un tesoro in un'isola dei Caraibi, sulle antiche rotte dei pirati. Provammo a scrivere qualche appunto: le idee non mancavano, anzi erano talmente buone che decidemmo di accantonare il soggetto per una futura avventura in grande stile (succede anche questo). E allora, che ambientazione usare? Per amore di varietà, non volevo un'ambientazione fantasy o gotica (tipo "Avventura nel castello"), anche perché non si presta molto ad avventure di piccole dimensioni, dove l'atmosfera non può essere sviluppata più di tanto. Occorreva anche motivare l'avventuriero: non c'è niente di peggio di un'avventura noiosa, in cui il giocatore si sente "staccato" dal gioco invece di "viverlo" davvero in prima persona. Un reattore nucleare impazzito? Troppo banale. Un computer maligno a bordo di

un'astronave (come HAL 9000 nel celebre "2001, odissea nello spazio")? Già più interessante, anche se difficile da sviluppare in poco spazio. Togliamo il computer e mettiamo il reattore impazzito a bordo dell'astronave. Quasi ci siamo. Aggiungiamo un tempo limite e 250 ignari passeggeri da salvare (in aggiunta alla propria pelle). Sì, così può andare.

La trama

Ora che abbiamo l'idea di base, vediamo di combinare una trama coerente. Il protagonista è il capitano dell'astronave, che dovrà riuscire a fermare in tempo il reattore. Naturalmente, dovrà cavarsela da solo: non deve poter scaricare su qualcun altro parte della responsabilità, altrimenti il gioco perde interesse e diventa un'avventura nella pubblica amministrazione. Ma non divaghiamo: dunque il comandante è solo ed i passeggeri sono, poniamo, beatamente addormentati.
Va bene che le moderne astronavi richiedono poco personale, ma è poco plausibile che basti una sola persona. Se poi stabiliamo che il comandante non è capace a fermare il reattore, occorre che qualcun altro lo sappia fare. D'altra parte, questo qualcun altro non dev'essere a portata di mano, altrimenti sarebbe in grado di risolvere il problema. Introduciamo allora un secondo pilota con mansioni tecniche, ma facciamo in modo che non sia accessibile. Potrebbe essersi suicidato (tipica sindrome da depressione spaziale), oppure essere stato vittima di un'incidente, magari a causa del reattore stesso.
In un'avventura spaziale che si rispetti, non può mancare un'uscita nel vuoto. Dobbiamo però motivarla in qualche modo. Ecco: il secondo pilota si trova all'esterno dell'astronave, e solo recuperandolo si può ottenere un'informazione essenziale. A questo punto, può essere ancora vivo: così il suo salvataggio fa parte della missione. Una volta salvato, riprende conoscenza (ma solo per un'attimo) e ci consegna qualcosa di fondamentale. Cosa? Vediamo... Il manuale di istruzioni che serve per disattivare il reattore. No, aggiungiamo un passaggio: la chiave del suo armadietto che contiene il manuale in questione e un camice antiradiazioni necessario per lavorare sul reattore.
A proposito, perché il secondo pilota stava all'esterno? Prendiamo a prestito un'idea da "2001": stava riparando l'antenna di comunicazione (così la nave è anche isolata dal resto dell'universo).
E con questo la trama è completa nelle sue linee essenziali: occorre equipaggiarsi per uscire nel vuoto (tuta e casco), uscire all'esterno, salvare il secondo pilota e procurarsi così la chiave, aprire il suo armadiet-

to e prendere camice e manuale, leggere quest'ultimo e disattivare il reattore.
Può andare, i dettagli li aggiungeremo in seguito.

La mappa

Ci resta da stabilire la disposizione dei vari locali che compongono l'astronave. Disegnamo una piantina di massima con lo stesso sistema che

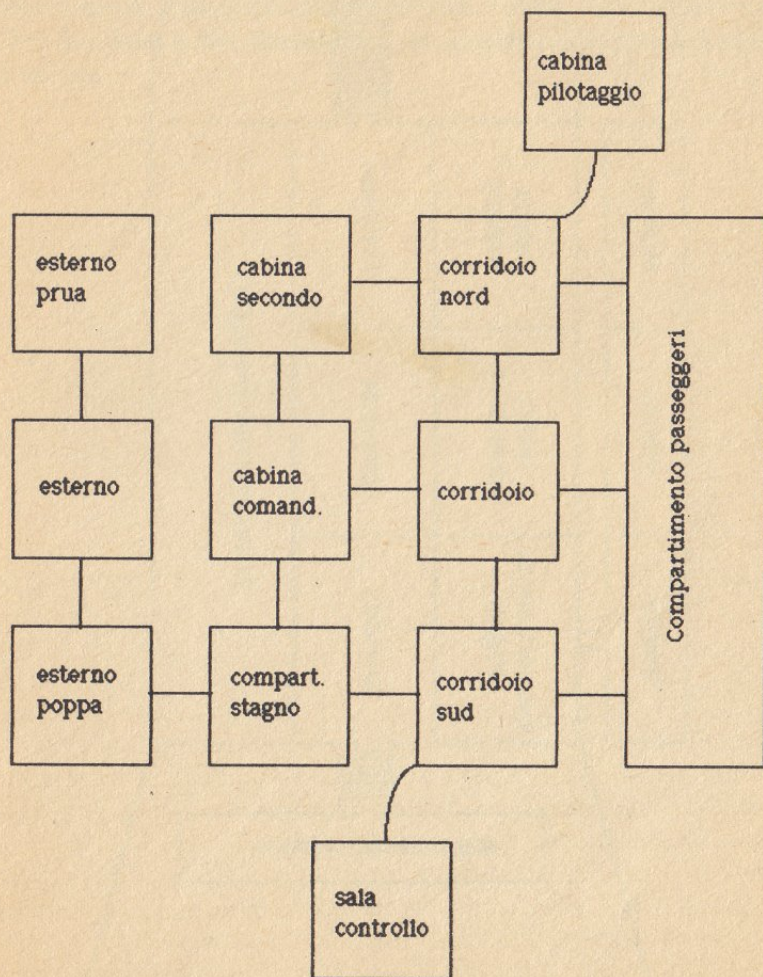



Fig. 1 — Disposizione dei locali ne "L'astronave condannata".

usiamo per mappare le avventure da risolvere: quadrati congiunti da linee che indicano le possibili direzioni. Per convenzione, il Nord è in alto, l'Est a destra (come nelle cartine geografiche). Per indicare un movimen-

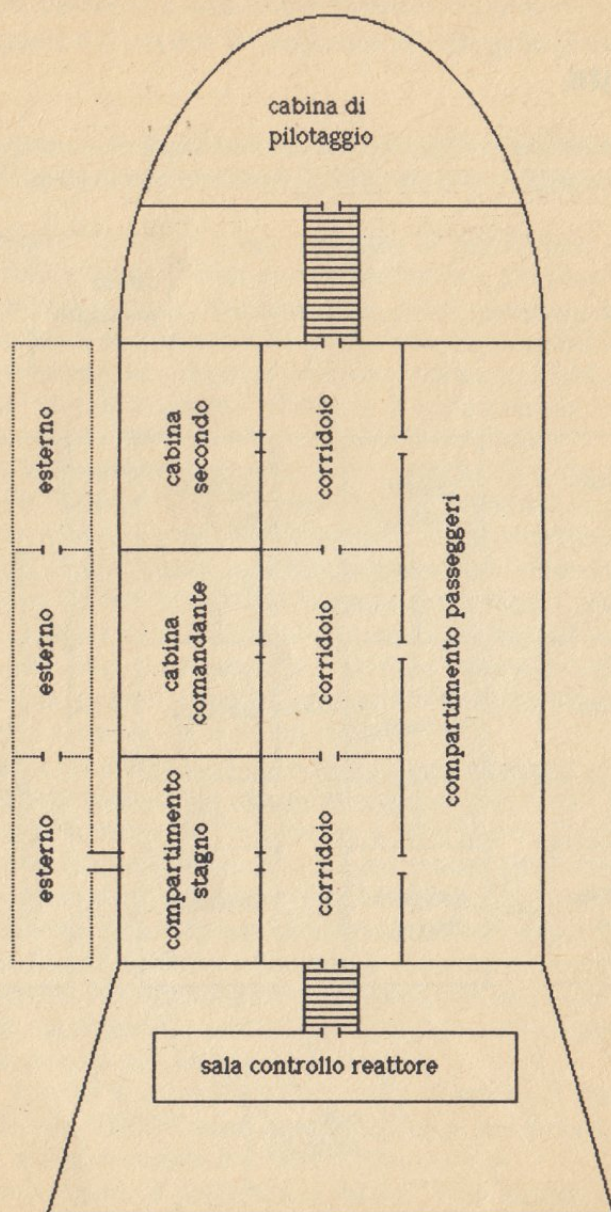
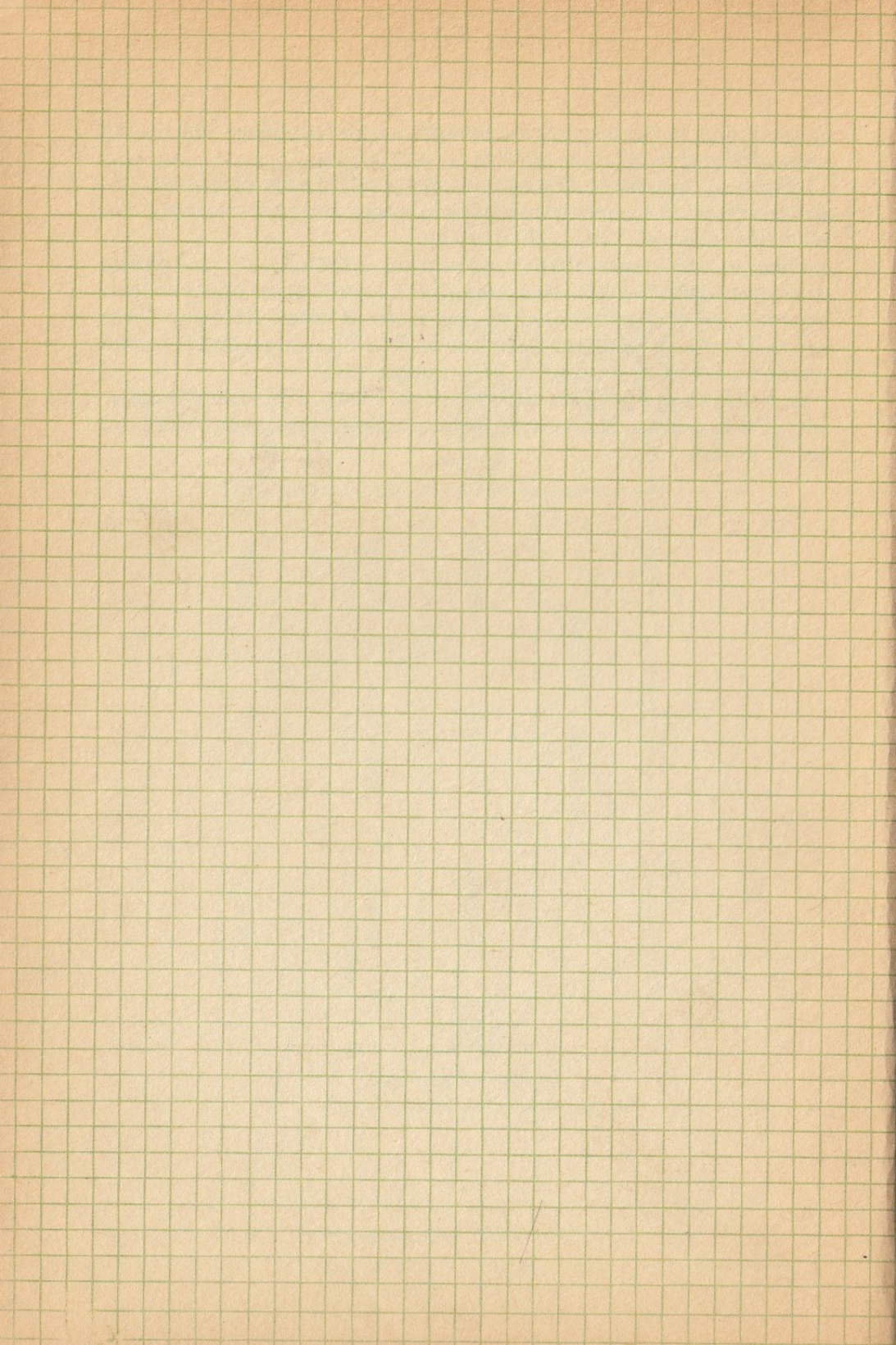
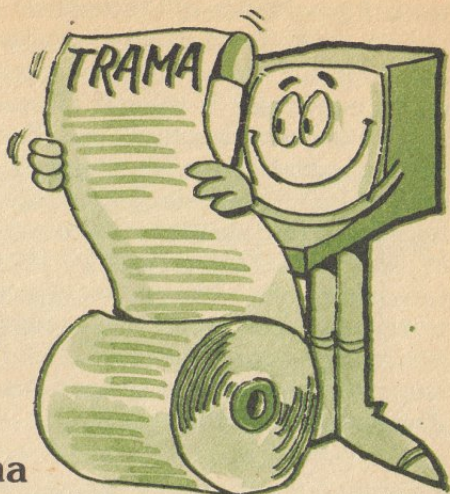



Fig. 2 — Pianta dell'astronave passeggeri "Neutronia".

to verso l'alto, facciamo partire la linea dall'angolo superiore destro, per il basso dall'angolo inferiore sinistro.
La Figura 1 mostra la disposizione dei locali, disegnata con questo sistema. Notate il compartimento stagno, necessario per recarsi all'esterno senza far uscire l'aria dall'astronave.
Sembra che vada bene, possiamo disegnarla in bella. La Figura 2 mostra la versione definitiva della stessa piantina. Le proporzioni reali non sono rispettate, ed abbiamo tralasciato vari locali di minor interesse, ma le severe Norme Di Sicurezza Galattica del 2313 ci proibiscono di fornire ulteriori particolari, sia pure di un'astronave civile.
A parte i dettagli, il progetto dell'avventura è finito: non resta che realizzarla. Per la cronaca, abbiamo impiegato (a parte il disegno in bella) poco più di un'ora.

COME NON SI SCRIVE UN'AVVENTURA

Dall'idea al programma

Ora che disponiamo di una trama, possiamo cominciare a scrivere il programma. Come si scrive un programma di questo genere? È lo stesso problema che mi trovai a fronteggiare nel 1981 quando, dopo aver giocato ad "Apple Adventure", decisi di scrivere un'avventura per conto mio (che poi divenne "Avventura nel castello"). A quei tempi ero totalmente disinformato, e non leggevo Byte (la principale rivista americana del settore), anzi ne conoscevo a malapena l'esistenza, per non dire di altre pubblicazioni meno note ma altrettanto interessanti (come il "Doctor Dobb's Journal Of Computer, Calisthenics & Orthodonthia", sì, proprio così) delle quali sfogliavo casualmente un numero ogni tanto. Un peccato, perché avrei potuto almeno conoscere quello che altri avevano fatto o stavano facendo, Scott Adams in testa. Dirò a mia parziale discolpa che ero soltanto un dilettante e che tali pubblicazioni non circolavano nemmeno in ambienti più qualificati, o che avrebbero dovuto essere tali (es. università), dove vigeva ancora intatto il culto del computer-dinosauro (a schede perforate, con lunghe code per poterlo usare).

Comunque, dovetti arrangiarmi. Per fortuna, avevo una discreta pratica di programmazione) e riuscii a tirar fuori, in un anno di lavoro nei momenti liberi, un programma che oggi, da professionista, posso ancora guardare senza vergognarmi. Date le premesse, è una bella soddisfazione. Il programma usato per scrivere "L'astronave condannata" deriva direttamente da quello di "Avventura nel castello", con molte semplificazioni ma anche con varie migliorie derivate da uno studio che ho compiuto nel 1984, e del quale avremo occasione di riparlare. La miglioria fondamentale, almeno dal vostro punto di vista, consiste nel linguaggio

usato: il programma di "Avventura nel castello" (nella sua prima versione) era scritto parte in Basic e parte in Assembly 6502 (il linguaggio macchina dell'Apple), con tutti i trucchi possibili per risparmiare memoria ed aumentare la velocità di esecuzione. Questo significa che era indissolubilmente legato allo specifico modello di computer sul quale era stato progettato. Il programma de "L'astronave condannata", invece, è tutto in Basic, ed in Basic "pulito". Quindi è facilmente portatile, cioè può funzionare con minime modifiche su macchine tra loro diverse come un PC IBM ed un Commodore 64. In compenso, non si presta particolarmente bene alla realizzazione di avventure di grandi dimensioni: d'altronde non si può avere tutto. Le semplificazioni che ho introdotto (e che riducono un poco l' "intelligenza" del programma) sono in gran parte volute, per concentrare la spiegazione sugli aspetti essenziali senza farsi distrarre dalle rifiniture.

Impostazioni sbagliate

Dicevo, poc'anzi, che sono tuttora soddisfatto dell'impostazione del mio primo programma di avventura. Lo sono perché, guardandomi in giro, vedo che la grande maggioranza delle avventure in circolazione sono costruite con tecniche molto più primitive e laboriose (l'unica eccezione che ho finora trovata sta nei giochi della Infocom, "Zork" in testa, al cui confronto il mio programma può andare a nascondersi). Anche in vari libri che insegnano "come scrivere un'avventura" vi sono impostazioni che obbligano il programmatore ad un lavoro da certosino, gran parte del quale potrebbe tranquillamente essere svolto da un programma scritto una sola volta per tutte. Perché questa è l'essenza stessa della programmazione dei calcolatori: scrivere un programma che risolva non un singolo problema, ma un'intera classe di problemi simili, al variare dei dati introdotti. Non avrebbe senso scrivere un programma per sommare 1+1, un altro per sommare 1+2, uno diverso per sommare 2+2, e così via: un solo programma può risolvere tutti questi problemi. Sembra ovvio, ma con l'aumentare della complessità e variabilità del problema la cosa diventa sempre meno facile, e richiede uno studio molto accurato. La tentazione di scrivere un programma che risolva solo il problema contingente diventa sempre più forte. Ad esempio, ho visto un gioco di avventura scritto più o meno in questo modo:

```
...
500 REM STANZA BIANCA
510 PRINT "SEI NELLA STANZA BIANCA. C'E' UN ARMADIO"
520 INPUT A$
```

```
530 IF A$="N" OR A$="NORD" GOTO 610:REM ROSSA
540 IF A$="E" OR A$="EST" GOTO 710:REM VERDE
550 PRINT "NON CAPISCO":GOTO 510
...
600 REM STANZA ROSSA
610 PRINT "SEI NELLA STANZA ROSSA. C'E' UN ARMADIO"
620 INPUT A$
630 IF A$="S" OR A$="SUD" GOTO 510:REM BIANCA
640 IF A$="APRI L'ARMADIO" THEN PRINT "E' VUOTO":GOTO 610
650 PRINT "NON CAPISCO":GOTO 610
...
```

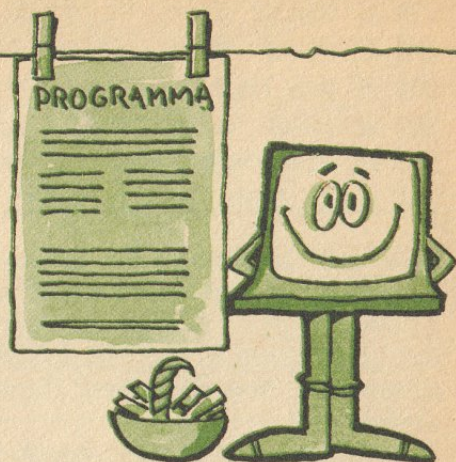
Avete capito come funziona? Per ogni luogo c'è un apposito gruppo di istruzioni, entro le quali il programma continua a girare fino a che l'avventuriero non decide di cambiare luogo. Ad esempio, le linee dalla 500 alla 550 si riferiscono al luogo chiamato "Stanza bianca". La linea 510 descrive il luogo, e la 520 attende il comando del giocatore (tipicamente ci sarà una subroutine invece di un'istruzione INPUT, ma il concetto è lo stesso). Se il giocatore ha scritto "N" o "NORD", parole entrambe valide per indicare uno spostamento verso nord, il programma salta al gruppo di istruzioni relative al luogo che si trova a nord della stanza bianca, nell'esempio un'ipotetica Stanza rossa (linea 610). Se ha scritto "E" oppure "EST", salta alle istruzioni relative alla "Stanza verde", e così via. Se il comando non è tra quelli previsti in quella stanza, la linea 550 ritorna all'inizio del ciclo (510).
Ogni stanza (luogo) ha il proprio ciclo di istruzioni, entro il quale sono ammesse soltanto certe parole o frasi. Per esempio, nel ciclo della stanza rossa (linee 610-650) sono ammesse "N" (o "NORD"), che torna al ciclo della stanza bianca, e "APRI L'ARMADIO", che stampa "È VUOTO" e torna all'inizio del ciclo. Lo stesso discorso vale per tutti i luoghi in cui il protagonista può muoversi. È una tecnica semplice e comprensibile.
Un'impostazione di questo tipo va evitata come la peste, e vi fornisco un elenco di buoni motivi:

- Il controllo delle direzioni possibili (IF A$="N" or A$="NORD"...) va ripetuto in ogni ciclo di luogo.
- Le frasi valide vanno scritte esattamente come sono previste dal programma. Se la frase "APRI L'ARMADIO" viene scritta con uno spazio in testa, in coda, o tra "APRI e L'ARMADIO", si ottiene un "NON CAPISCO". Lo stesso succede se viene trascurato l'articolo o manca l'apostrofo.

- Ci sono notevoli problemi con le frasi valide in tutti i luoghi, ad esempio le azioni su un oggetto trasportabile.
- Dovendo scrivere un'altra avventura, il programma va riscritto da cima a fondo.

Potrei continuare, ma mi sembra sufficiente. In sostanza, occorre un grande lavoro per ottenere un programma non troppo soddisfacente e che occupa inutilmente molta memoria. Peggio ancora, il gioco si presenta con un comportamento decisamente "stupido", nel senso che accetta solo ed esclusivamente le frasi previste nel luogo in cui si trova l'avventuriero (che corrisponde ad un "luogo" fisico del programma, inteso come gruppo di istruzioni). Come magra consolazione, gira velocemente.
Quello che vi ho presentato è un esempio limite (reale, purtroppo), ma gran parte delle avventure in circolazione presentano problemi dovuti ad un cattivo progetto del programma, che si traduce poi in una perdita di realismo, e dunque di interesse del giocatore. Tipico commento: - Ma non capisce un accidente, questa stupida macchina? — Dove già l'illusione di vivere un'avventura in un mondo fantastico è andata a farsi benedire, con buona pace degli sforzi dell'autore.
È chiaro che la qualità di un'avventura dipende da chi l'ha scritta, ma è altrettanto chiaro che la fantasia non basta: senza tecniche adeguate è difficile realizzare un gioco di questo genere che sia capace di "catturare" i giocatori fornendo un livello accettabile di simulazione. I prossimi capitoli dimostrano come sia possibile ottenere questo con una struttura compatta, facile da usare e riutilizzabile per diverse avventure.

CAPITOLO 5

UNA MACCHINA A STATI

Una routine universale

La soluzione migliore non è quella di un programma che esegue istruzioni diverse a seconda del luogo mostrata come esempio negativo nel precedente capitolo), ma consiste in un programma che esegua sempre lo stesso ciclo fondamentale, cambiando soltanto il valore di alcune variabili e, di conseguenza, i messaggi stampati su video. Un esempio di questa impostazione è stato presentato da Gianni Giaccaglini nel volume "Stendere un programma come si deve", pubblicato in questa stessa collana.

VERIFICARE

Per chi non l'avesse letto (male!), ed anche per gli altri, propongo un esempio simile: un mini-programma (o meglio: una routine, cioè una parte di programma) che stampa la descrizione della stanza in cui si trova l'avventuriero e gli consente di muoversi con il solito sistema, cioè scrivendo la direzione in cui vuole andare. Ecco la routine:

```
100 REM INIZIALIZZA VARIABILI
...
200 REM CICLO
210 PRINT DE$(LU):REM DESCRIZIONE
220 INPUT A$:D=0
230 IF A$="N" OR A$="NORD" THEN D=1
240 IF A$="S" OR A$="SUD" THEN D=2
250 IF A$="E" OR A$="EST" THEN D=3
260 IF A$="O" OR A$="W" OR A$="OVEST" THEN D=4
```

```
270 IF A$="A" OR A$="ALTO" OR A$="SALI" THEN D=5
280 IF A$="B" OR A$="BASSO" OR A$="SCENDI" THEN D=6
290 IF D=0 THEN PRINT"NON CAPISCO":GOTO 210
300 NL=MA(LU,D):REM NUOVO LUOGO
310 IF NL=0 THEN PRINT "NON PUOI":GOTO 210
320 LU=NL:GOTO 210:REM MUOVE
```

Tutto qui. Se all'inizio (tra la linea 100 e la 200) introduciamo gli opportuni dati negli elementi degli array DE$() e MA(), questa routine di 13 linee fa tutto il lavoro necessario per tener conto degli spostamenti dell'avventuriero su una mappa comunque complessa e di qualunque dimensione. Non ci credete? Passiamo subito ad una dimostrazione pratica.
Supponiamo che la nostra mappa sia composta di tre luoghi, chiamati "Stanza rossa", "Stanza bianca" e "Stanza verde", disposti come in Figura 1. La Figura 2 mostra quale dev'essere il contenuto dei due array

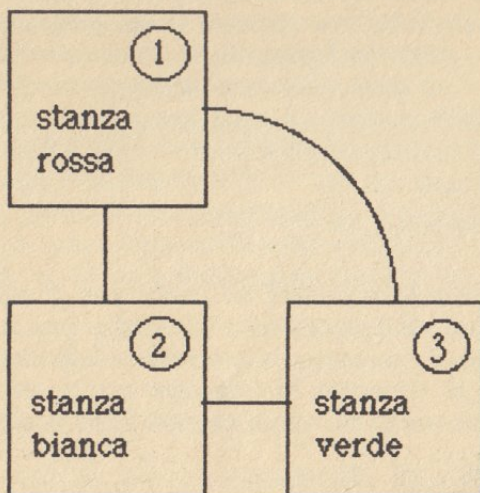



Fig. 1 — Esempio minimo di mappa per avventura.

DE$() e MA(). Il primo, di tipo alfanumerico (stringa), contiene semplicemente le descrizioni dei luoghi, in ordine di numero. Quindi, DE$(1) è la descrizione del luogo 1, DE$(2) è la descrizione del luogo 2 e DE$(3) è la descrizione del luogo 3. Come avete brillantemente dedotto, i luoghi vanno numerati a partire da 1.
Torniamo per un attimo al programmino, e precisamente alla linea 210. Questa stampa il contenuto di DE$(LU), cioè la descrizione del luogo

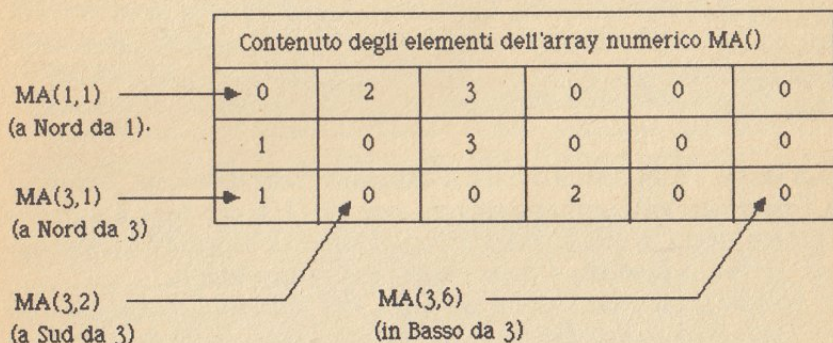
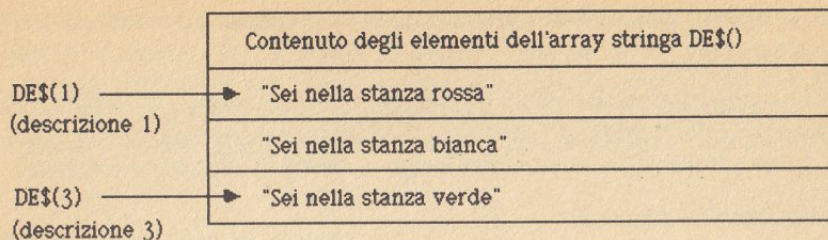



Fig. 2 — Array di mappa e loro contenuto.

numero LU. LU contiene quindi il numero del luogo corrente (cioè quello in cui si trova l'avventuriero). Supponiamo di aver messo LU=1 all'inizio del programma: significa che il personaggio si trova nel luogo 1, e quindi la linea 210 stampa DE$(1), cioè "Sei nella stanza rossa", che è appunto la descrizione del luogo 1 (DE sta per "descrizione", è sempre meglio usare per le variabili nomi che aiutino a ricordarne il significato). A questo punto dovreste aver capito che per spostare l'avventuriero, cioè per cambiare luogo, è sufficiente cambiare il valore di LU. Se infatti LU vale 2, la linea 210 stampa "Sei nella stanza bianca", e se vale 3 stampa "Sei nella stanza verde". Il giocatore ha l'illusione di cambiare luogo, ma per il programma cambia semplicemente il valore di una variabile, e precisamente LU. Le istruzioni eseguite rimangono sempre le stesse.
D'accordo, ma come facciamo a cambiare LU a seconda della parola scritta dal giocatore, in modo che gli spostamenti siano coerenti con la nostra mappa? È molto semplice: ad ogni luogo sono associati sei numeri, corrispondenti alle sei direzioni, nell'ordine da noi stabilito (stabiliamo, ad esempio, l'ordine Nord, Sud, Est, Ovest, Alto, Basso).

Questi numeri sono contenuti in una riga della matrice, o array a due dimensioni, MA() (MA sta per "mappa").

Sempre con riferimento alla Figura 2, i sei numeri contenuti in MA(1,1), MA(1,2) MA(1,3), MA(1,4) MA(1,5), MA(1,6) sono i numeri dei luoghi in cui si va a finire spostandosi dal luogo 1 verso la direzione indicata.

Numeriamo le direzioni:

- Nord=1
- Sud=2
- Est=3
- Ovest=4
- Alto=5
- Basso=6

Se dal luogo 1 il giocatore decide di muoversi nella direzione 3 (cioè ad est), l'elemento MA(luogo,direzione), cioè MA(1,3) contiene il numero del nuovo luogo, in questo caso 3. Sè poniamo LU uguale a questo numero, abbiamo spostato il personaggio nel nuovo luogo.
Non sempre lo spostamento è possibile. Per indicare l'impossibilità di movimento in una certa direzione da un certo luogo, mettiamo uno zero nell'elemento corrispondente. Ad esempio, dal luogo 1 non si può andare ad ovest, perciò MA(1,4) contiene zero.
Riassumendo, ogni riga della matrice corrisponde ad un luogo, ed ogni colonna ad una direzione. L'incrocio di riga e colonna indica se è possibile lo spostamento da quel luogo in quella direzione e, in caso affermativo, il numero del luogo in cui si va a finire. Se LU è il luogo attuale, e D il numero corrispondente alla direzione scelta, MA(LU,D) è il nuovo luogo, o zero se il movimento non è possibile.
Torniamo ad esaminare il programma. La linea 220 attende il comando del giocatore e pone D a zero (nessuna direzione introdotta). Le linee dalla 230 alla 280 servono per mettere in D il codice (numero) della direzione scritta, senza alcuna pretesa (per ora) di impiegare il metodo più efficiente. Ad esempio, se il giocatore ha scritto "EST" la linea 250 esegue un D=3 (codice di "Est"). Se la parola introdotta non corrisponde ad alcuna direzione, D rimane zero.
La linea 290 controlla se è stata introdotta una direzione valida. In caso contrario (e lo si vede dal fatto che D vale zero), stampa "NON CAPISCO" e rimanda all'inizio del ciclo.
La linea 300 consulta la mappa contenuta nella matrice MA() per conoscere il luogo adiacente al luogo corrente (LU) nella direzione indicata (D), e mette in NL il numero del nuovo luogo. Non lo mette subito in LU,

perché potrebbe essere zero (spostamento impossibile), ed in questo caso andrebbe perso il valore di LU.
Se lo spostamento non è valido, la linea 310 stampa "NON PUOI" e torna all'inizio del ciclo (LU non è cambiato), altrimenti la linea 320 cambia il valore di LU, spostando così il personaggio nel nuovo luogo.

Programma e dati

Il concetto fondamentale, che sta alla base di tutto il meccanismo, è questo: cambiano i dati, ma non cambia il programma. Volendo infatti rappresentare una mappa del tutto diversa, basta cambiare il contenuto dell'array di descrizioni DE$() e della matrice di collegamenti MA(). Il programma rimane identico: il lavoro è stato fatto una volta sola. Come vantaggio secondario, ma non trascurabile, il programma occupa molto meno spazio.
Un programma di questo tipo è detto **table driven**, cioè "pilotato da una tavola", in quanto il suo comportamento dipende esclusivamente dai dati contenuti in un'apposita tabella, cioè la matrice MA(). È anche detto **macchina a stati**, in quanto la differenza tra una situazione e l'altra (tra un luogo e l'altro) dipende esclusivamente dallo stato (valore) di una o più variabili, nel nostro caso soltanto da LU (i puristi mi perdonino la descrizione non troppo formale). Il programma gira sempre nello stesso ciclo, dovunque si trovi l'avventuriero.
Rimane il problema di come introdurre i valori nelle tavole, cioè negli array DE$() e MA().
Si potrebbe fare in questo modo:

```
110 DE$(1)="Sei nella stanza rossa"
120 DE$(2)="Sei nella stanza bianca"
130 DE$(3)="Sei nella stanza verde"
140 MA(1,1)=0
150 MA(1,2)=2
160 MA(1,3)=3
...
```

Si vede subito che questa non è la soluzione più pratica: è un enorme spreco di memoria per eseguire istruzioni ripetitive (pensate ad una mappa con un centinaio di stanze). C'è una tecnica decisamente migliore, basata su una comodissima istruzione del Basic, che tra l'altro manca in alcuni linguaggi "più evoluti": l'istruzione DATA, che consente di introdurre nel programma un elenco di dati numerici o di stringhe.

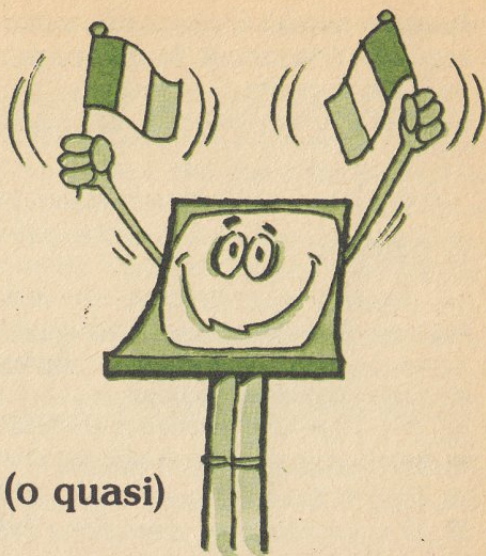
Possiamo dunque scrivere le nostre due tavole in questo modo:

```
300 DATA "Sei nella stanza rossa"
310 DATA 0,2,3,0,0,0
320 DATA "Sei nella stanza bianca"
330 DATA 1,0,3,0,0,0
340 DATA "Sei nella stanza verde"
350 DATA 1,0,0,2,0,0
```

Alla descrizione di ogni luogo, fanno immediatamente seguito i passaggi possibili nelle sei direzioni. Per leggere questi dati, facciamo uso dell'istruzione READ. Questa equivale alla INPUT, ma legge dalla lista di DATA invece che dalla tastiera (partendo dal punto a cui era arrivata la READ precedente). Ecco come si riduce l'inizializzazione (preparazione iniziale) delle variabili per il programma mostrato in precedenza:

```
110 NL=3:REM NUMERO LUOGHI
120 DIM DE$(NL),MA(NL,6)
130 FOR LU=1 TO NL
140 READ DE$(LU)
150 FOR D=1 TO 6:READ MA(LU,D):NEXT D
160 NEXT LU
170 LU=1
```

Notate che la linea 110 specifica il numero di luoghi, che è un valore fisso, cioè una costante. È buona regola indicare le costanti una sola volta all'inizio del programma, invece che disperderle nel programma stesso. Così le eventuali (in realtà sicure) modifiche sono molto più semplici ed è meno facile commettere errori. Ancora più intelligente sarebbe far leggere il numero di luoghi dai DATA con un READ NL: il programma resterebbe inalterato al variare del numero di luoghi.
La linea 120 stabilisce le dimensioni delle nostre tavole, cioè i due array DE$() e MA(). In molti Basic questo non è necessario se le dimensioni non superano il 10, come è il nostro caso, ma indicarlo comunque non fa male. Il ciclo aperto alla linea 130 e chiuso alla 160 si ripete per ogni luogo, quindi tre volte nell'esempio. La linea 140 legge la descrizione del luogo nell'elemento corrispondente dell'array stringa DE$(), mentre il ciclo alla 150 legge la mappa delle direzioni nell'array bidimensionale, o matrice, MA. La linea 160, infine, stabilisce che il personaggio inizia il gioco nel luogo 1.

IL COMPUTER CAPISCE L'ITALIANO

Intelligenza artificiale (o quasi)

Molti giocatori di adventure si chiedono con stupore come diavolo faccia il computer a capire le frasi scritte dal giocatore. Altri non ci badano nemmeno, come fosse ovvio che il computer sia dotato di una certa facoltà di comprensione. È o non è una macchina intelligente?
La risposta è: NO, il computer non è una macchina intelligente. Il computer è stupido come una vacca. Anzi, ripensandoci, è ben più stupido di una vacca: mi scusino le pacifiche frequentatrici dei pascoli d'alta quota per questo inappropriato ed offensivo paragone. Il computer è invece il soldato perfetto, eterno sogno dei generali di tutte le epoche: dotato di notevoli poteri, obbedisce ciecamente e senza emozioni a qualunque ordine ricevuto, senza chiedersene i motivi e senza prevederne le conseguenze. Esegue tutto alla lettera, senza minimamente comprendere quello che sta facendo e senza chiedersi se il significato letterale delle istruzioni ricevute corrisponde realmente a quello voluto dal programmatore (caso assai raro). L'intelligenza non è nel computer, ma nel programma. Quest'ultimo, è il caso di ricordarlo, è soltanto una sequenza di istruzioni per la macchina stupida, scritte da un programmatore più o meno sveglio. L'intelligenza che il calcolatore dimostra, insomma, è ancora quella di chi ha scritto il programma. Il vantaggio del computer è che, con un po' di tecnica, è possibile scrivere un programma adatto a risolvere non solo uno specifico problema, ma tutti i problemi dello stesso tipo, risparmiandosi così parecchio lavoro. Nel capitolo precedente ho fornito un esempio di questa possibilità, mostrandovi un programma che "capisce" il concetto di direzione. Se avete letto bene, vi sarà chiaro che il programma non capisce un accidente, ma si limita a

spostare numeri e riprodurre scritte secondo regole ben definite (la sequenza di istruzioni che forma il programma). La definizione di queste regole è, appunto, il compito del programmatore che vuole dare alla macchina un'apparenza di comportamento intelligente.

Il parser

Bisogna dunque scrivere un programma capace di simulare la comprensione dei concetti che stanno alla base dei giochi di avventura, e che fondamentalmente riguardano:

- Mappa, luoghi e spostamenti.
- Oggetti fissi ed oggetti trasportabili.
- Comprensione ed esecuzione delle azioni dell'avventuriero.

Il primo problema (mappa e spostamenti) è già risolto. Il secondo si può facilmente affrontare associando ad ogni oggetto una stringa (la sua descrizione) ed un numero (il luogo in cui si trova). Tornerò più avanti sui dettagli. Il problema più complesso è decisamente il terzo: capire la frase scritta alla tastiera e modificare di conseguenza lo stato del gioco.
Un programma che effettua l'analisi di una frase (o, più in generale, di una sequenza di simboli) sulla base di un insieme di regole (o sintassi) è detto **parser**, o analizzatore sintattico. Un gioco di avventura consiste sostanzialmente in un parser che, sulla base delle parole introdotte e delle loro associazioni, decide il successivo comportamento del programma. Per scrivere questo programma, occorre prima definire le regole che ne governano il comportamento. Per fare questo, cerchiamo di classificare le possibili frasi scritte dal giocatore:

- Una direzione (spostamento), es. "NORD".
- Un'azione espressa con una sola parola, es. "INVENTARIO".
- Un'azione su un oggetto, es. "PREMI IL BOTTONE".

Per ora non entro in ulteriori dettagli, rimandandoli al capitolo dedicato alle azioni. Noto soltanto che una stessa azione può avere effetti diversi in luoghi diversi (es. "Leggi il cartello"), e che l'oggetto citato può essere anche assente. Occorre tener conto di queste ed altre possibilità, se si vuole realizzare un gioco con un buon livello di simulazione.

Analisi lessicale

Per facilitare il lavoro al parser (o analizzatore sintattico), occorre separare le parole che compongono la frase (trascurando eventuali spazi). Inoltre, la manipolazione delle parole come stringhe costa parecchio spazio: è più pratico assegnare un numero (codice) ad ogni vocabolo e passare al parser soltanto i codici delle parole che compongono la frase. Bisogna anche scartare le parole errate o comunque non riconosciute (es. "Prindi"). Questi lavori di bassa manovalanza sono compito dell'**analizzatore lessicale,** la routine che riceve una stringa contenente la frase introdotta dal giocatore, la separa in vocaboli distinti e restituisce i codici dei vocaboli costituenti la frase. A volte l'analizzatore lessicale svolge anche un lavoro che, strettamente parlando, sarebbe di competenza del parser: riconosce e tralascia gli articoli, dato che non alterano il significato della frase e caricherebbero il parser di un lavoro inutile. Ad esempio, se il giocatore ha scritto:

"RIPARA L'ANTENNA"

la routine di analisi lessicale separa le tre parole "RIPARA", "L" (l'apostrofo viene considerato come uno spazio separatore), e "ANTENNA". L'articolo "L" viene scartato, e vengono restituiti i codici numerici corrispondenti a "RIPARA" (27) e "ANTENNA" (69). Sarà poi compito del parser decidere se questa coppia di codici corrisponde ad una frase valida nel luogo e nella situazione corrente dell'avventuriero. Dato che il nostro parser è alquanto semplice, ogni parola successiva alla seconda (esclusi articoli) viene semplicemente ignorata. Non sono quindi possibili frasi come "Guarda dietro al cartello" e simili.

Interprete e dati

Siamo quasi pronti a partire, i dettagli possiamo esaminarli strada facendo. Dal prossimo capitolo, si comincia a scrivere l'avventura. Vi riassumo come funziona la tecnica impiegata per "L'astronave condannata":

- Il giocatore introduce una frase.
- L'analizzatore lessicale la separa in parole e ne ricava i codici.
- Il parser decide se i codici (le parole) indicano un'azione valida.
- Se sì, viene chiamata la routine Basic che esegue l'azione corrispondente.

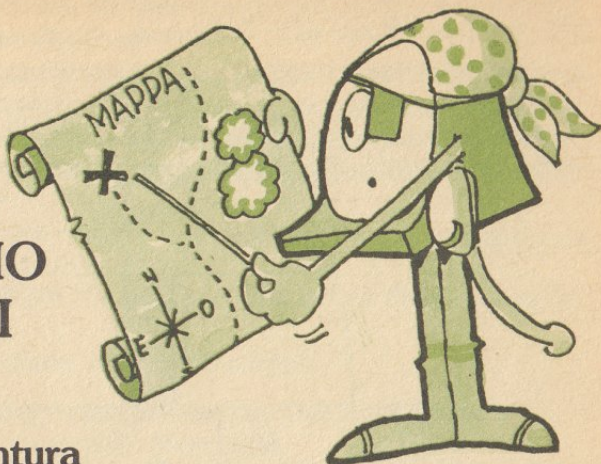
Non ho citato le direzioni: anche queste, in effetti, costituiscono

un'azione come tutte le altre, e non c'è alcuna convenienza nel trattarle diversamente.
Il programma finale è composto da una collezione di dati, da una parte variabile (le azioni Basic) e da una parte fissa (analizzatore lessicale, parser e routine connesse). Questa parte fissa è perfettamente reimpiegabile per altre avventure, senza necessità di modifica. Dato che interpreta una serie di dati forniti dal programmatore, ho chiamato **interprete** questa sezione riutilizzabile del programma. Anche alcune delle routine Basic fondamentali (es. quelle usate per direzioni, "Save", "Prendi",ecc.) sono riutilizzabili tali e quali o con poche modifiche.
Il programma si compone dunque di tre parti:

- **L'interprete,** riusabile senza modifiche per altre avventure.
- **I dati** dell'avventura (vocaboli, mappa, descrizioni, ecc.).
- **Le routine Basic** delle azioni (alcune delle quali riutilizzabili).

Una buona notizia: per scrivere un'avventura non c'è bisogno di comprendere il funzionamento dell'interprete, che costituisce la parte decisamente più complessa del programma. Per chi fosse interessato darò comunque, nell'ultima parte del libro, una dettagliata descrizione tecnica del suo funzionamento.

CAPITOLO 7

MAPPA, DIZIONARIO E OGGETTI

La vostra avventura

Scopo di questo libro, è di mettervi in grado di scrivere una vostra avventura. Riferendomi al programma de l'astronave condannata, ricordo che l'interprete, costituito dalle linee fino alla 1160, può essere riutilizzato senza modifiche (a parte l'aggiustamento di qualche GOSUB che chiama vostre routine).
Da questo capitolo, inizio a descrivere le varie fasi della costruzione di un'avventura, usando come esempio quella che avete giocato. Al termine, per gli interessati, descriverò anche il funzionamento dell'interprete.

La mappa

Una volta stabilita la disposizione dei luoghi in cui si svolge l'avventura ed i relativi collegamenti, non resta che introdurre questi dati nel programma. Come ho detto nel precedente capitolo, si tratta soltanto di scrivere una lista di istruzioni DATA.
La piantina dei luoghi in cui si svolge "L'astronave condannata" è riportata in Figura 1, con l'aggiunta dei numeri che identificano ciascun luogo. Ci sono 11 luoghi, numerati da 1 a 11.
Per ciascun luogo, occorre fornire al programma una descrizione del luogo stesso ed i numeri dei sei luoghi confinanti (vedi Capitolo 5). Con riferimento al listato (riprodotto nel Capitolo 2), le linee dalla 4130 alla 4230 contengono queste informazioni, in ordine di luogo (il luogo 1 alla linea 4130, e via di seguito).

Notate che le descrizioni sono incomplete: per risparmiare memoria, ho omesso il punto finale ed il "Sei" iniziale, che viene aggiunto automatica-

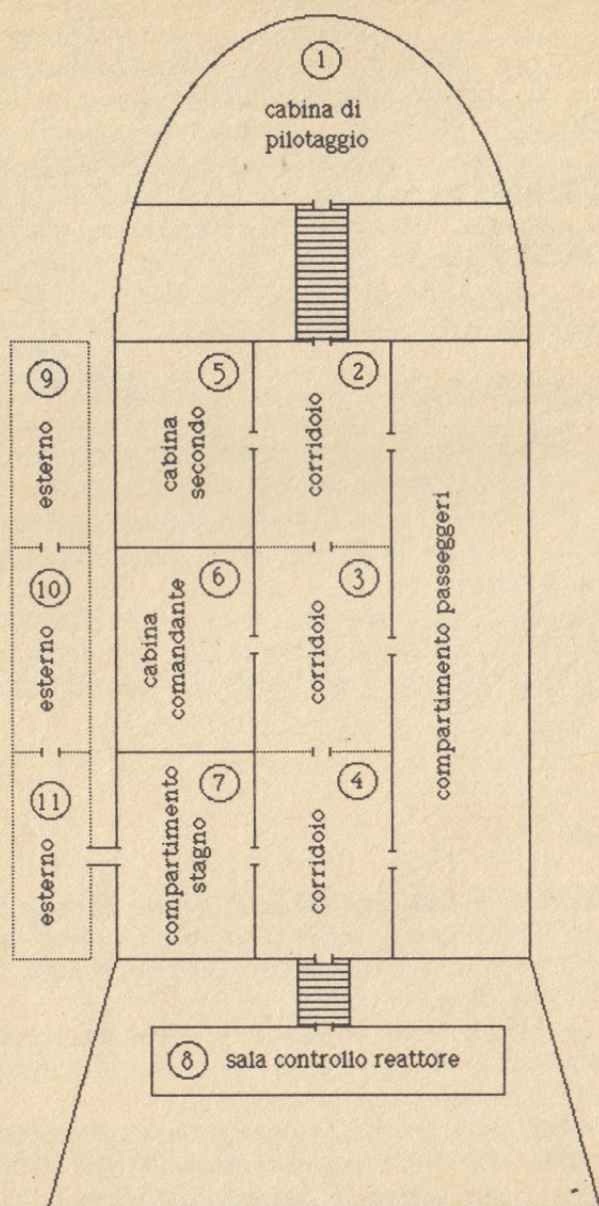



Fig. 1 — Pianta dell'astronave passeggeri "Neutronia".

mente dall'interprete al momento di stampare la descrizione del luogo (linea 760, se ci tenete a saperlo).
Inoltre, i numeri che rappresentano i luoghi adiacenti nelle sei direzioni non sono scritti separatamente, ma raccolti in una sola stringa di 12 caratteri (sempre per risparmiare memoria). I primi due caratteri indicano il luogo adiacente a Nord, i due successivi quello a Sud, e di seguito Est, Ovest, Alto e Basso.
Dato che occorre indicare sempre due caratteri, i numeri di una sola cifra vanno preceduti da uno zero.
Al termine delle linee di DATA, un FM indica la fine delle mappa (in fondo alla linea 4230).
Riassumendo, per introdurre una vostra mappa:

- Disegnate accuratamente la piantina, con tutti i collegamenti.
- Numerate i luoghi con numeri progressivi.
- Per ogni luogo, scrivetevi una descrizione ed i numeri dei luoghi adiacenti nelle sei direzioni, nell'ordine N,S,E,O,A,B. Ad esempio:

  (Sei) nel corridoio,2,4,0,6,0,0

Ricordo che lo zero indica una direzione non ammessa. Poi riscrivete i sei numeri di seguito, estendendoli a due cifre se necessario:

02 04 00 06 00 00

E, togliendo gli spazi, ottenete la stringa:

"020400060000"

La descrizione e questa stringa, separate da una virgola, formano la linea di DATA che serve all'interprete (confrontare con la linea 4150).

- Infine, non dimenticate la stringa "FM" che indica all'interprete la fine della mappa. Potete scriverla su una riga di DATA isolata o, più convenientemente, alla fine dei DATA dell'ultimo luogo (linea 4230). Se la dimenticate, otterrete errori strani alla partenza del programma (magari su tutt'altra linea, quando l'interprete perde il filo nel leggere i DATA).

Il dizionario

Definita la mappa, occorre scrivere tutti i vocaboli che dovranno essere "capiti" dall'interprete. Tanto più ricco il vocabolario, tanto più sarà facile simulare un comportamento "intelligente" del gioco. Per fare un

confronto, "L'astronave condannata" usa una cinquantina di vocaboli, mentre "Avventura nel castello" ne accetta circa 400.
Il lavoro risulta semplificato se si definisce fin dall'inizio il maggior numero possibile di vocaboli, aggiungendo poi soltanto quelli relativi a varianti o piccole migliorie del gioco. Cominciamo dunque ad elencare i termini necessari, associando a ciascuno un numero (a nostro piacimento) che l'interprete userà al posto del vocabolo (con grande risparmio di spazio).
Per una migliore organizzazione, è comodo suddividere le parole in quattro gruppi:

- Direzioni e comandi generali, usabili in più avventure (es. "Nord", "Prendi", "Save").
- Verbi intransitivi e transitivi (es. "Salta", "Apri").
- Oggetti trasportabili (es. "Chiave").
- Oggetti fissi e di arredamento (es. "Armadietto").

La Figura 2 mostra i vocaboli usati nell'avventura spaziale, con i relativi

L'astronave condannata: dizionario

Codice	Parola	Codice	Parola	Codice	Parola
1	N,Nord	2	S ,Sud	3	E ,Est
4	O,Ovest,W	5	A,Alto,Sali	6	B,Basso,Scendi
7	Il,Lo,La,L	8	Prendi	9	Lascia
10	Guarda	11	Save	12	Load
13	Cosa,Inventario				
20	Indossa,Metti	21	Togli	22	Apri
23	Tira	24	Spingi	25	Leggi
26	Premi,Schiaccia	27	Ripara,Aggiusta		
50	Casco	51	Tuta	52	Camice
53	Secondo	54	Chiave	55	Manuale
60	Cartello	61	Pulsante,Bottone	62	Verde
63	Giallo	64	Rosso	65	Leva
66	Indicatore	67	Armadietto	68	Letto
69	Antenna	70	Etichetta		

Fig. 2 — Vocaboli del dizionario, in ordine di codice.

codici. Ho riservato i codici inferiori a 20 per i termini di uso generale (se non cambiate questi codici, vi facilitate parecchio la vita). Ho poi suddiviso le restanti tre categorie, facendo partire i codici da numeri arbitrari (20, 50 e 60). I colori (codici 62, 63, 64) servono per frasi tipo "Premi il verde" o "Premi il rosso", e sono dunque trattati come oggetti fissi, in analogia con "Premi il bottone". Gli articoli (da ignorare) sono raccolti sotto lo stesso codice (7). I sinonimi (che hanno lo stesso codice) sono scritti sulla stessa riga (es. "O" ed "Ovest").

I codici sono semplicemente numeri arbitrari compresi tra 1 e 98, senza alcuna implicazione di ordine o spazio occupato. Una suddivisione come quella mostrata è comunque comoda, per distinguere i vari tipi di parole.

Sorge ora un problema tecnico: la ricerca nel dizionario è troppo lenta. Quando l'analizzatore lessicale ha isolato una parola (es. "ANTENNA") e chiama la routine che consulta il dizionario per trovarne il codice (il numero che abbiamo appena assegnato), questa routine deve far passare tutti i vocaboli fino a trovare la parola cercata. Se questa è in fondo alla lista, o non c'è, può passare un tempo relativamente lungo, e comunque fastidioso per chi gioca.

La migliore soluzione consiste nel tenere il dizionario in ordine alfabetico, in modo da poter usare una veloce routine di ricerca binaria (cfr. ABC, capitolo 20). Dato che l'ordinamento impiega un certo tempo, è meglio inserire i vocaboli già in ordine alfabetico. Questo vuol dire che il tempo risparmiato dal giocatore si paga con un po' di lavoro in più del programmatore (di solito conviene: dopotutto, si fa una volta sola). Quindi, al lavoro: il risultato finale si vede in Figura 3. Naturalmente, i sinonimi sono ora sparsi in vari punti della lista. Quando mettete in ordine il dizionario, verificate bene: **il corretto ordinamento alfabetico è fondamentale.** Altrimenti, qualche vocabolo potrebbe non essere riconosciuto. Se avete dei dubbi... consultate un dizionario!

Tornando al programma, le linee di DATA dalla 3990 alla 4090 contengono il dizionario, in ordine alfabetico. Ogni parola è seguita dal suo codice, ed un "FD" finale indica all'interprete la fine del dizionario.

Riassumendo:

- Elencate i vocaboli, suddivisi per gruppi.
- Assegnate un codice ad ogni vocabolo o gruppo di sinonimi.
- Riordinate i vocaboli in ordine alfabetico, mantenendo vicino a ciascuno il suo codice.
- Riportate il tutto sotto forma di DATA nel programma.

In ordine alfabetico:

Parola	Codice	Parola	Codice	Parola	Codice
A	5	Aggiusta	27	Alto	5
Antenna	69	Apri	22	Armadietto	67
B	6	Basso	6	Bottone	61
Camice	52	Cartello	60	Casco	50
Chiave	54	Cosa	13	E	3
Est	3	Etichetta	70	Giallo	63
Guarda	10	Il	7	Indicatore	66
Indossa	20	Inventario	13	L	7
La	7	Lascia	9	Leggi	25
Letto	68	Leva	65	Lo	7
Load	12	Manuale	55	Metti	20
N	1	Nord	1	O	4
Ovest	4	Premi	26	Prendi	8
Pulsante	61	Ripara	27	Rosso	64
S	2	Sali	5	Save	11
Scendi	6	Schiaccia	26	Secondo	53
Spingi	24	Sud	2	Tira	23
Togli	21	Tuta	51	Verde	62
W	4				

Fig. 3 — Vocaboli del dizionario, in ordine alfabetico.

Gli oggetti

L'avventuriero incontra oggetti di vario genere. Alcuni di questi possono essere presi e trasportati (come una chiave), altri fanno semplicemente parte dell'arredamento (come un letto). Tutti possono essere riferiti o manipolati (almeno con "Guarda"). Gli oggetti trasportabili possono trovarsi in vari luoghi, o in possesso dell'avventuriero, o provvisoriamente fuori del gioco.
L'interprete richiede una lista degli oggetti, con indicate le varie caratteristiche. Gli oggetti vanno numerati, associando a ciascuno di essi un numero in ordine progressivo a partire da 1. Questo numero non ha alcun rapporto con il codice nel dizionario del nome dell'oggetto, ma serve all'interprete (ed al programmatore) per riferirsi all'oggetto in questione. L'ordine è arbitrario, ma va tenuto presente che gli oggetti vengono descritti al giocatore (vedo...) in ordine di numero.

Ad ogni oggetto vanno associati una stringa (la sua descrizione) ed altri due numeri: il codice nel dizionario della parola che lo descrive, ed il luogo in cui si trova. La Figura 4 mostra l'elenco degli oggetti usati ne "L'astronave condannata".

L'astronave condannata: oggetti

Num	Oggetto	Codice	Luogo
1	un indicatore	66	-1
2	un pulsante	61	-1
3	un'etichetta	70	-1
4	una tuta	51	1
5	un cartello bianco	60	-2
6	un cartello giallo	60	-4
7	un letto	68	-5
8	un armadietto	67	-5
9	un letto	68	-6
10	un armadietto	67	-6
11	un casco	50	6
12	un pulsante rosso	61	-7
13	un pulsante verde	61	-7
14	un indicatore	66	-8
15	una leva	65	-8
16	un pulsante rosso	61	-8
17	un pulsante verde	61	-8
18	un cartello rosso	60	-8
19	un pulsante giallo	61	-8
20	il secondo pilota	53	9
21	un'antenna parabolica	69	-9
22	un camice	52	-99
23	una chiave	54	-99
24	un manuale	55	-99

Fig. 4 — Gli oggetti dell'avventura.

Il luogo contiene anche ulteriori informazioni:

- Se il luogo è zero, l'oggetto è trasportato dall'avventuriero.
- Se il luogo è negativo, l'oggetto non è prendibile (es. il letto). Questo è un tipico trucco per tenere due informazioni diverse (luogo e prendibilità) nello stesso numero.
- Se il luogo è —99, l'oggetto è nel "limbo", cioè fuori gioco.

Ad esempio, l'oggetto numero 6 è descritto come un cartello giallo, nel dizionario ha codice 60 ("Cartello"), si trova nel luogo 4 (fine corridoio sud) e non è prendibile (il numero di luogo è negativo). Possono esserci più oggetti con lo stesso nome (e quindi con lo stesso codice di dizionario), ma devono essere tutti non prendibili (es. i letti 7 e 9, ed i pulsanti 2, 12, 13, 16, 17, 19), in modo che non possano mai trovarsi nello stesso luogo (il programma non si confonde troppo, ma il giocatore sì).
Nella lista di DATA, gli oggetti sono elencati nello stesso modo, in ordine di numero e terminando con "FO". Il numero in sè non è indicato: il primo oggetto elencato ha numero 1, il secondo ha numero 2, e così via.
Le linee dalla 4400 alla 4510 contengono l'elenco degli oggetti usati nell'avventura (identico a quello di Figura 4). Riassumendo:

- Elencare gli oggetti usati, ed assegnare a ciascuno un numero progressivo.
- Stabilire la descrizione di ciascun oggetto (es. un' "antenna parabolica").
- Scrivere il codice della parola nel dizionario (es. il codice di Antenna, cioè 69).
- Scrivere il luogo in cui l'oggetto si trova all'inizio del gioco (es. 9 per l'antenna), o —99 se è fuori gioco, o zero se è trasportato. Mettere il segno meno se l'oggetto non è prendibile (come nel caso dell'antenna, il cui luogo diventa dunque —9).
- Riportare il tutto nei DATA.

È importante tenere ben segnato l'elenco degli oggetti, con il relativo numero progressivo. Infatti, l'interprete crea un array LO%() contenente il luogo di ciascun oggetto, in ordine di numero. Ricordo che il simbolo %, percento, indica che si tratta di un array di numeri interi: serve soltanto a risparmiare spazio (e, in alcuni Basic, anche tempo nei calcoli). **Questo array LO%() contiene quasi tutto lo stato dell'avventura,** se si eccettuano la posizione dell'avventuriero (contenuta nella variabile LU) ed alcune informazioni accessorie. Tutti gli altri dati sono infatti costanti (non cambiano nel corso del gioco). Agendo su LO%(), il programmatore può muovere gli oggetti, o trasformarli. Ad esempio, l'istruzione:

```
LO%(23)=7
```

muove l'oggetto 23 (la chiave) nel luogo 7 (il compartimento stagno). Dato che la descrizione degli oggetti visibili è automatica, l'istruzione citata fa "apparire" la chiave, che può successivamente essere presa, guardata, ecc. Con la stessa tecnica si può trasformare un oggetto, ad esempio facendo sparire una spia verde (mettendola nel luogo —99) ed

apparire una spia rossa (mettendola nel luogo corrente, LU, o meglio — LU, dato che si tratta di un oggetto non prendibile). Dal punto di vista del giocatore è sempre lo stesso oggetto, ma il programmatore lavora in realtà con due oggetti distinti.
La manipolazione del luogo degli oggetti attraverso LO%() è lo strumento principale a disposizione del programmatore.
Altri esempi:

```
LO%(6)=0
```

porta l'oggetto 6 (il casco) in possesso dell'avventuriero, mentre:

```
LO%(24)=LU
```

mette l'oggetto 24 (il manuale) nel luogo corrente. Se era in un'altra stanza, "appare"; se era in possesso del personaggio, viene "lasciato" per terra. Infine, l'istruzione:

```
IF LO%(23)=0 THEN...
```

significa: se l'oggetto 23 (la chiave) è nel luogo zero (trasportato), cioè se l'avventuriero possiede la chiave... (segue la conseguenza, ad esempio si può aprire l'armadietto).

AZIONE!

Eseguire un'azione

Siamo arrivati al punto più delicato dell'intero meccanismo: l'esecuzione degli ordini dati alla tastiera. È qui che si vede la "intelligenza" del programma (o la sua stupidità): vediamo dunque il problema in dettaglio. Il giocatore scrive una frase composta da una o due parole significative (esclusi quindi gli articoli), che chiamiamo "prima parola" (o "verbo") e, se esiste, "seconda parola" (o "nome"). Il parser deve, sulla base dei codici di queste parole, chiamare la routine che esegue l'azione desiderata.

Dovrebbe essere ormai abbastanza chiaro cosa significhi eseguire un'azione: si tratta di stampare un messaggio e/o modificare lo stato di alcune variabili, tipicamente LU (il luogo dell'avventuriero) e LO%() (il luogo di uno o più oggetti). Per esempio, la frase "Prendi il manuale" deve far partire una routine Basic che controlla se il manuale è presente (e quindi prendibile) e, in caso affermativo, sposta il manuale stesso nel luogo zero (trasportato) e stampa il messaggio "Fatto".

Non è però conveniente scrivere routine Basic distinte per "Prendi il manuale", "Prendi la tuta", "Prendi il camice", ecc. Una sola routine può occuparsi del verbo "Prendi" in generale.

Questa è in genere la tecnica usata per scrivere giochi di avventura, tecnica che è anche illustrata in vari libri sull'argomento: il parser chiama la routine Basic corrispondente alla prima parola della frase (il verbo), lasciando a questa routine il compito di distinguere tra i vari casi particolari. Il "selettore" della routine da eseguire funziona più o meno in questa maniera:

```
500 IF C1 = 1 THEN GOSUB 1000
510 IF C1 = 2 THEN GOSUB 1010
520 IF C1 = 3 THEN GOSUB 1020
...
```

Se C1 è il codice della prima parola della frase, il funzionamento è chiaro: la subroutine che inizia alla linea 1000 esegue le azioni relative al verbo con codice 1 (ad esempio "Prendi"), la 1010 quelle relative al verbo con codice 2, e così di seguito. Se i codici partono da 1 e sono progressivi, si usa un sistema molto più semplice ed efficiente:

```
500 ON C1 GOSUB 1000,1010,1020, ...
```

che ha esattamente lo stesso effetto. Se C1 vale 1, viene eseguita la subroutine numero1, cioè la prima dell'elenco (equivale ad un GOSUB 1000), e così via.

Problemi di comprensione

Come dicevo, quello illustrato è il sistema di gran lunga più usato. Probabilmente il motivo sta nella sua semplicità: il parser non deve prendere decisioni, ma lascia tutto il lavoro alle routine chiamate. Come spesso capita, si tratta di un'impostazione sbagliata. La semplicità del parser si paga con una maggiore complicazione delle routine chiamate (che sono tante, mentre il parser è uno solo) oppure, a scelta, con un comportamento più stupido del programma. Consideriamo, ad esempio, la frase "Premi il manuale". Ci sono varie possibilità:

- Il manuale è presente nello stesso luogo del personaggio.
- Il manuale si trova in un altro luogo, o fuori gioco.
- Il manuale è già in possesso dell'avventuriero.
- Il manuale non è un oggetto prendibile.

Se consideriamo queste possibilità nella routine di "Prendi", dobbiamo fare un lavoro analogo in molte altre routine ("Lascia", "Guarda", ecc.). Se non le consideriamo, la simulazione ed il realismo vanno a farsi benedire. Di solito, viene usata una soluzione di questo tipo: si considera che la seconda parola sia un oggetto, e si verifica che questo sia presente. Ma anche questa "soluzione" presenta i suoi problemi. Consideriamo la frase "Cerca il manuale": ovviamente il manuale non è presente, dunque il parser risponde "Non capisco" (mentre si vorrebbe poter stam-

pare una frase appropriata, del tipo "Cercatelo da te, io non ne ho voglia"). In questo caso, il controllo di presenza è fastidioso. Una situazione del genere si presenta, ne "L'astronave condannata", con i pulsanti colorati: la frase "Prendi il verde" va eseguita anche se non esiste nella stanza l'oggetto "Verde", ma solo l'oggetto "Pulsante". D'altra parte, l'azione "Leggi il manuale" va eseguita soltanto se il manuale è presente.

Di conseguenza, molte avventure commerciali vanno in crisi già con frasi relativamente semplici, come "Guarda l'albero" (ad esempio rispondendo "Non noto nulla di particolare" anche se l'albero non è presente), o in altre situazioni analoghe (molte rispondono sistematicamente "Non capisco").

Potrei continuare con l'elenco dei problemi, delle soluzioni parziali e delle relative trappole e complicazioni, ma preferisco fermarmi a questo punto ed illustrare una soluzione più flessibile e che, a spese di una leggera complicazione del parser (che ho scritto io, una volta per tutte) risparmia lavoro e problemi ai programmatori (cioè a voi).

La tavola delle azioni

Il parser de "L'astronave condannata" non è particolarmente furbo, anche se è già decisamente sopra la media di quelli contenuti nelle avventure commerciali e di quelli presentati nei libri che insegnano "come scrivere un'avventura". Un parser migliore è usato in "Avventura nel castello" (tra l'altro, tratta diversamente i verbi transitivi dagli intransitivi), ed uno ancora più evoluto è contenuto nel mio studio su un linguaggio specializzato per la scrittura di avventure (ADL-1). Un parser di altissimo livello si trova invece nei giochi della Infocom; ad esempio "Zork" riesce a "capire", con un'eccellente illusione di realismo, frasi come "Prendi la sfera di cristallo e mettila sul supporto di diamante" (naturalmente in presenza di più sfere e diversi supporti!).

Tornando al mio parser, funziona in questo modo: il programma consulta una **tavola delle azioni,** che contiene informazioni sulle azioni valide, e in base a questa decide se eseguire una routine Basic, e quale. Le routine Basic sono numerate a partire da 1 e sono chiamate con un'istruzione ON...GOSUB, come già visto in precedenza. La tavola può riconoscere quattro tipi di frasi composte da due parole:

1) Una coppia di parole (verbo+nome) in un dato luogo (es. "Premi il pulsante" nella sala controllo reattore).

2) Una parola specifica (verbo) seguita da una qualunque altra, in un dato luogo (es. "Guarda XXX" in una stanza con illusioni ottiche).
3) Una coppia di parole (verbo+nome) in un luogo qualunque (es. "Leggi il manuale", dovunque).
4) Una parola specifica (verbo) seguita da una qualunque altra, in un luogo qualunque (es. "Prendi XXX", dovunque).

In tutti questi casi, è possibile specificare se l'oggetto eventualmente indicato dalla seconda parola dev'essere presente perché la routine Basic venga eseguita, o se l'azione è valida comunque, anche in assenza dell'oggetto. Sono riconosciuti anche due tipi di frasi composte da una sola parola, che si riconducono in pratica ai tipi 1 e 2 con la seconda parola assente:

5) Una parola (verbo) in un certo luogo (es. "Est" nel corridoio).
6) Una parola (verbo) in un luogo qualunque (es. "Inventario").

Cosa molto importante, la ricerca nella tavola avviene nell'ordine in cui ho elencato le varie possibilità. Se il giocatore ha scritto "EST", il parser cerca se è prevista la parola "Est" nel luogo corrente dell'avventuriero (caso 5). Se la trova, esegue l'azione indicata. Se non la trova, cerca se è prevista la parola "Est" valida dovunque (caso 6). La trova e chiama la routine indicata (quella di spostamento). Se non la trovasse, stamperebbe "Non capisco". Lo stesso succede nel caso di due parole: prima il parser guarda se è previsto il caso 1 (quel verbo e quel nome in quel luogo), e solo se non lo trova passa ai casi 2, 3, 4, e infine stampa "Non capisco". La regola è che il caso particolare ha la precedenza su quello generico. In altri termini: di solito la parola "Est" ha un certo effetto (muove verso est), ma in un certo specifico luogo voglio che chiami una routine particolare.
Non è tutto chiaro? Beh, me l'aspettavo. È ora di fare qualche esempio che dissipi la fitta nebbia nella quale temo di avervi avvolti. Facciamolo subito.

Direzione: Nord

Cominciamo dal caso più semplice: le direzioni. Ho detto in precedenza che le direzioni non richiedono alcun trattamento speciale: sono parole come le altre, si tratta soltanto di far eseguire la routine adeguata. Questa routine sarà unica per tutte le direzioni, e deciderà il da farsi in base al codice della prima (ed unica) parola, codice contenuto nella variabile C1

(con un po' di fantasia, potete indovinare che C2 contiene il codice della seconda parola).
La tavola delle azioni deve dunque contenere queste informazioni:

- Se la frase contiene solo la parola N o sinonimo (codice 1), esegui la routine Basic numero 1 (che si occupa delle direzioni).
- Se la frase contiene solo la parola "S" o sinonimo (codice 2), esegui la routine Basic numero 1 (sempre la stessa).
- Se la frase contiene solo la parola E o sinonimo (codice 3), esegui la routine Basic numero 1 (idem, come sopra).

E così di seguito per le altre tre direzioni. Ci saranno dunque sei informazioni nella tavola delle azioni. Ogni informazione consiste in due numeri: la **sintesi della frase** ed il **numero della routine** da eseguire. La sintesi della frase è un numero che si calcola in questo modo:

LU*10000 + C1*100 + C2

Detto più alla buona: le prime due cifre indicano il luogo in cui la frase ha effetto (zero se vale dovunque), le due successive sono il codice della prima parola e le ultime due sono il codice della seconda parola (zero se manca). Quindi la sintesi della frase "NORD", valida dovunque, si costruisce così

Luogo: 00(dovunque, si può tralasciare)
Codice1: 01(codice di "NORD" o "N")
Codice2: 00(nessuna parola)

Il risultato, o sintesi della frase (ottenuto accostando i tre dati) è 000100. Dato che si tratta di un numero e non di una stringa, è più banalmente 100. Nella tavola delle azioni, che il programma si legge all'inizio dalle linee di DATA, è inutile sprecare spazio con zeri non significativi, quindi per far eseguire la routine numero 1 in risposta alla frase "NORD" basta scrivere:

```
DATA 100,1
```

Allo stesso modo, la sintesi di SUD è 0200, quella di EST 0300, di "OVEST" 0400, di "ALTO" 0500, di "BASSO" 0600. Dato che tutte queste frasi devono far chiamare la routine numero 1, la tavola delle azioni conterrà sei informazioni, ciascuna composta di sintesi frase e numero della routine da eseguire:

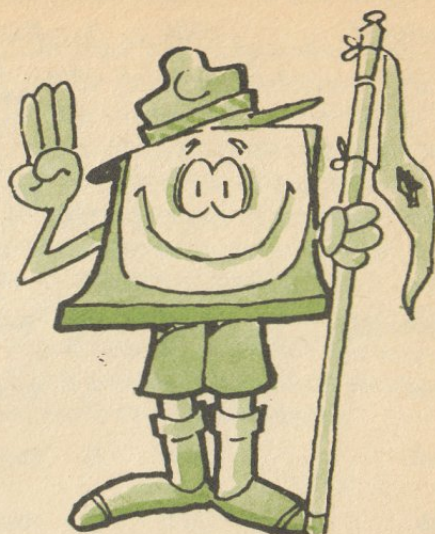
```
DATA 100,1,200,1,300,1,400,1,500,1,600,1
```

Guardate la linea 4270 del programma: inizia esattamente con questi

dati. Per accelerare la ricerca delle azioni, la tavola va scritta in ordine di numero di sintesi frase. Per ora non ci sono problemi, ma è importante non dimenticarlo.
Quindi, se il giocatore scrive la frase "NORD" nel luogo 14:

- L'analizzatore lessicale restituisce i codici delle due parole: C1=1 (codice di "Nord") e C2=0 (codice di parola nulla).
- Il parser compone la sintesi di LU, C1 e C2, cioè numero del luogo corrente, codice della prima parola, codice della seconda parola. Ottiene 140100 e cerca questo numero nella tavola delle azioni. Supponiamo che non lo trovi. Vuol dire che l'azione "Nord" non è prevista nel luogo 14.
- Il parser compone allora la sintesi di C1 e C2, cioè 0100, cioè 100, e la cerca nella tavola delle azioni. La trova: è un'azione prevista come valida in tutti i luoghi. Legge il numero dell'azione da eseguire: 1.
- Il parser chiama l'azione numero 1, usando il selettore delle azioni (linea 1710), che chiama la subroutine alla 1780, cioè la routine Basic di direzione (finalmente!).

Per non lasciare misteri insoluti, andiamo a vedere cosa succede nella famigerata routine che esegue gli spostamenti (linee 1780-1800). Succede esattamente quello che vi ho già mostrato nel capitolo 5, con la sola differenza che il numero del nuovo luogo non viene letto da una matrice numerica ma da un array di stringhe. DI$(LU) non è altro che la stringa di 12 cifre che è stata introdotta di seguito alla descrizione del luogo numero LU (vedi capitolo 7, mappa, e linee 4130-4230). Il MID$ alla linea 1780 isola le due cifre corrispondenti alla direzione scelta, basandosi sul codice C1 (le prime due per "Nord", le due seguenti per "Sud", ecc.), ed il VAL alla stessa linea le converte finalmente in un numero: il numero del nuovo luogo, che va provvisoriamente in A. Se la direzione non è permessa, la 1790 lo fa notare, altrimenti la 1800 esegue lo spostamento. Noto che la linea 1780 funziona solo se i codici di dizionario delle sei direzioni sono 1, 2, 3, 4, 5, 6. Quindi, non cambiateli!

UN PO' DI BUONE AZIONI PER DARE L'ESEMPIO

Si salvi chi può

In Figura 1 potete vedere i miei appunti relativi alle azioni generiche de "L'astronave condannata". Hanno un aspetto ordinato soltanto perché li

L'astronave condannata: azioni generiche

```
N               0100
S               0200
E               0300
O               0400
A               0500
B               0600     1    muove

Prendi XXX      0899    -2    if luogo(OG)=0
                                 "Già fatto"
                              else if OG non prendibile
                                 "Non è possibile"
                              else if OG=Tuta and luogo(Camice)=0
                                 "Togli prima il camice"
                              else if OG=Camice and luogo(Tuta)=0
                                 "Togli prima la tuta"
                              else
                                 luogo(OG)=0
                                 if OG=Casco,Tuta o Camice
                                    "Ora l'hai addosso"
                                 else
                                    "Fatto"
```

Lascia XXX	0999	3	if OG=0 or luogo(OG)<>0 "Non ce l'hai" else if LU>=9 {fuori} luogo(OG)=-99 "Perso nello spazio" else luogo(OG)=LU "Fatto"
Guarda XXX	1099	-4	"Nulla di particolare"
Save	1100	5	salva situazione
Load	1200	6	rilegge situazione
Cosa	1300	7	stampa inventario

Fig. 1 — L'astronave condannata: azioni generiche.

ho scritti con il Macintosh, anziché con carta e matita (è più comodo). Ho pensato di proporveli senza ritoccarli, anche se contengono qualche imprecisione e non sempre sono perfettamente coerenti, perché mi sembrano adatti a mostrare il paziente lavoro che sta dietro le quinte di un'avventura, anche una molto semplice come questa.

Gli appunti sono organizzati in questo modo: in prima colonna sono scritte le frasi che il programma deve riconoscere (raggruppate se devono far eseguire la stessa routine, come le sei direzioni), in seconda colonna c'e la sintesi numerica della frase (eventuale numero luogo, codice prima parola, codice seconda parola), segue il numero dell'azione (routine Basic) da eseguire, infine c'è una descrizione sintetica dell'azione stessa. Le azioni di Figura 1 sono quelle generiche, cioè valide in tutti i luoghi ed indipendenti dalla specifica avventura (salvo alcune eccezioni, che vedremo).

Cominciamo con tre azioni semplici: "Save", "Load" e "Inventario", rispettivamente "registra la situazione corrente" (altamente consigliata prima di intraprendere azioni avventate), "riprendi una situazione precedente" ed "elenca gli oggetti che possiedo" (con il comodo sinónimo "Cosa").

Il meccanismo di chiamata di queste azioni funziona esattamente come quello delle direzioni, facendo eseguire l'adatta routine Basic di azione (rispettivamente 5, 6 e 7). Vi ricordo che i codici delle parole si trovano nel dizionario preparato in precedenza (capitolo 7). Le routine Basic di

"Save" e "Load" si trovano alle linee 2000 e 2060, e dipendono naturalmente dal computer usato. Sono molto primitive, nel senso che è permesso il salvataggio di una sola situazione, con nome fisso ("ASTRO", definito alla linea 1010). Sono sicuro che sapete fare di meglio. La routine di "Inventario" (linea 2120) fa invece uso di una subroutine dell'interprete, quella che descrive gli oggetti presenti in un certo luogo. Dato che gli oggetti posseduti dall'avventuriero sono nel luogo zero, tanto vale riutilizzare la stessa routine. Vi conviene lasciare tale e quale questa azione. Per inciso, il GOTO 590 al termine della linea 2120 è concettualmente un GOSUB 590 seguito da un RETURN. L'effetto è identico, e si risparmia tempo (in pratica la routine di inventario ritorna usando il RETURN che sta al termine della routine chiamata).

Guarda chi si vede

L'azione "Guarda XXX" è invece più complessa: dev'essere eseguita in risposta ad una frase composta dalla parola "Guarda" seguita da una qualunque parola valida, sempre in qualunque luogo. La sintesi della frase si costruisce come prima:

Luogo:	00 (dovunque, si può tralasciare)
Codice1:	10 (codice di "GUARDA"
Codice2:	99 (una qualunque parola valida)

Una novità: indicando 99 come codice della seconda parola, si intende "una qualunque parola valida, purché non nulla". La sintesi della frase è dunque 1099 e, dato che dobbiamo eseguire l'azione 4, nella tavola dovremmo scrivere:

```
DATA 1099,4
```

Invece, nella tavola si trova:

```
DATA 1099,—4
```

(circa a metà della linea 4290). Questa è un'ulteriore comodità: se il numero della routine è negativo, significa che il parser deve controllare che la seconda parola si riferisca ad un oggetto **presente o trasportato,** cioè nel luogo corrente od in possesso dell'avventuriero. In caso contrario, stampa "Qui non c'è" e si rifiuta di chiamare la routine indicata. Dato che per guardare un oggetto, quest'ultimo deve ovviamente essere presente, mettiamo il segno meno e ci assicuriamo così che la routine nu-

mero 4 venga eseguita solo se il giocatore guarda un oggetto realmente visibile. Come potete vedere alla linea 1970, la routine numero 4 si limita a stampare "Non noto nulla di particolare". Si tratta infatti della routine generica di "Guarda...", che è in fondo alla lista di precedenza (caso 4 tra quelli elencati nel capitolo 8) e serve a dare comunque una risposta, in mancanza di azioni più interessanti (come "Guarda l'indicatore"). In gergo tecnico, è una routine **di default,** cioè da eseguire in difetto (in mancanza) di altre routine con precedenza più alta.

Prendere o lasciare

E siamo arrivati a due azioni fondamentali nei giochi di avventura: prendere e lasciare gli oggetti. Cominciamo dalla prima.
Volendo un minimo di realismo, nell'azione "Prendi" bisogna considerare che:

- L'oggetto citato dev'essere presente.
- L'oggetto non dev'essere già in possesso dell'avventuriero.
- L'oggetto dev'essere prendibile.

La prima condizione è automatica: basta mettere un segno meno davanti al numero dell'azione (vedi infatti la figura 1: il numero dell'azione da eseguire è —2). Per soddisfare le altre due, la routine di "Prendi" va impostata più o meno in questo modo:

- Se l'oggetto è già posseduto, stampare "Già fatto" e rientrare.
- Altrimenti, se l'oggetto è di tipo non prendibile (luogo negativo, vedi capitolo 7), stampare "Non è possibile" e rientrare.
- Altrimenti, spostare l'oggetto nel luogo zero (trasportato), stampare "Fatto" e rientrare.

Per riferirsi all'oggetto citato nella frase, si usa la variabile OG, nella quale l'interprete mette il numero dell'oggetto (non il codice di dizionario! Vedi capitolo 7) citato come seconda parola. **Attenzione:** se l'oggetto non è presente (nel luogo LU) o trasportato, OG contiene zero e non va usata. Perciò, LO%(OG) è il luogo dell'oggetto citato come seconda parola. Io ho l'abitudine di tenere i miei appunti in una forma di "pseudocodice", cioé in una specie di linguaggio strutturato del quale vi fornisco subito un esempio riscrivendo la routine di "Prendi":

```
    if luogo(OG)=0
        "Già fatto"
```

```
else if OG non prendibile
      "Non è possibile"
else
      luogo(OG)=0
      "Fatto"
```

Non c'è poi molta differenza rispetto a prima: è solo più sintetico. Per prima cosa, le parole **se** e **altrimenti** sono sostituite dalle equivalenti inglesi **if** ed **else**. Il vantaggio sta nella brevità e nella somiglianza con istruzioni Basic, che semplifica poi il lavoro di traduzione in programma. Inoltre, scrivo semplicemente tra virgolette un riassunto dei messaggi da stampare e faccio ampio uso dell'**indentazione,** che consiste nello spostare verso destra (indentare) le operazioni (istruzioni) relative ad un certo **if,** per vederle più chiaramente. Dato che questo libro non è un corso di programmazione strutturata, mi fermo qui. Penso comunque che i miei appunti siano abbastanza leggibili.
Tornando alla routine di "Prendi", si può tradurla in Basic in questo modo:

```
1830 IF LO%(OG)=0 THEN PRINT "Gia' fatto":RETURN
1840 IF LO%(OG)<0 THEN PRINT "Non e' possibile":RETURN
1870 LO%(OG)=0
1890 PRINT "Fatto":RETURN
```

Notate come la routine corrisponda esattamente agli appunti. In particolare, la linea 1840 verifica che l'oggetto citato sia prendibile controllando che il suo luogo non sia negativo (se è negativo, significa per convenzione "non prendibile", vedi capitolo 7).

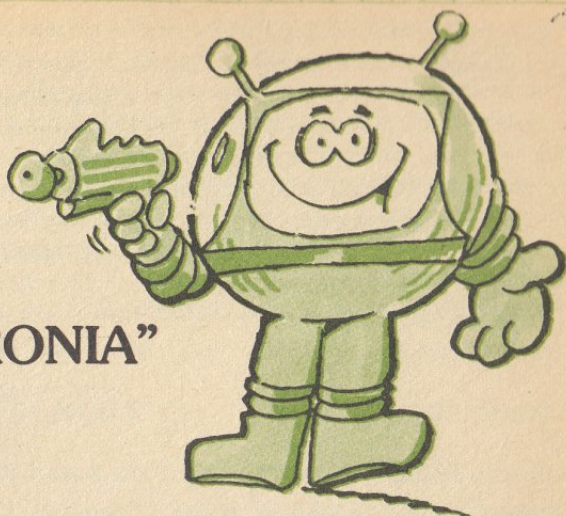
La routine di "Prendi" usata ne "L'astronave condannata" è leggermente più complessa, perché tiene conto di alcuni casi particolari: impedisce di mettere la tuta se si ha già addosso il camice (e viceversa), e stampa "Ora l'hai addosso" invece di "Fatto" se il personaggio prende il casco, la tuta o il camice. Verificate queste differenze sugli appunti di Figura 1 e nelle linee 1830-1890 (in pratica, sono state aggiunte le linee 1850, 1860 e 1880). In una vostra avventura, basta togliere queste linee aggiunte per avere una routine generica di "Prendi".

Passiamo all'azione reciproca: quella di lasciare un oggetto. Qui il lavoro è più semplice: se l'oggetto non è trasportato, non è lasciabile. Dato che vogliamo stampare "Non ce l'hai" se l'oggetto non è trasportato (che sia presente o meno), bisogna evitare il controllo preliminare dell'interprete

(che stamperebbe "Qui non c'è" in caso di assenza). Per evitare il controllo, basta usare un numero di azione positivo (infatti è 3, e non —3). Ma c'è una trappola: a questo punto non basta guardare LO%(OG) per sapere dove sta l'oggetto. Infatti, come ho detto prima, se l'oggetto non è presente o trasportato OG vale zero e LO%(OG) è teoricamente, un errore. In realtà, LO%(0) contiene zero (questo elemento non è mai stato usato), ma ciò non significa affatto che l'oggetto sia trasportato, significa solo che stiamo guardando una variabile priva di significato! Insomma, occorre anche controllare che OG non valga zero, altrimenti vuol dire che l'oggetto non è certamente lasciabile. L'azione di "Lascia" è quindi:

```
if OG=0 or luogo(OG)< >0
        "Non l'hai"
else
      luogo(OG)=LU
      "Fatto"
```

Dovreste sapere che **or** significa **oppure,** altrimenti ristudiatevi il Basic. Come per l'azione "Prendi", anche per la "Lascia" c'è un caso particolare che riguarda "L'astronave condannata": se il numero del luogo è maggiore o uguale a 9 (cioè 9, 10 o 11) ci si trova nello spazio esterno, e qualunque oggetto lasciato viene perso (spostandolo nel luogo —99, il "limbo"). Le linee di programma dalla 1920 alla 1940 sono la routine di "Lascia".
Come in tutti i programmi, non c'è un modo solo per risolvere i problemi. Si poteva anche tenere una routine più pulita per "Lascia" e prevedere la frase "LASCIA XXX" nei luoghi 9, 10, 11. La mia soluzione è più corta, ma forse meno consigliabile a chi ama le cose a tutti i costi ben ordinate
Lo stesso discorso vale per "PRENDI IL CASCO", "PRENDI LA TUTA" e "PRENDI IL CAMICE" dove però, volendo separare le tre azioni speciali, le cose si facevano un poco più complesse. Occorreva, in pratica, ripetere quattro volte i controlli della routine di "Prendi". Probabilmente non ne valeva la pena.

A BORDO DEL "NEUTRONIA"

Azioni su misura

Le azioni finora viste sono quelle di tipo generico, riutilizzabili per ogni avventura. Si tratta ora di scrivere le azioni specifiche di una particolare avventura, e precisamente de "L'astronave condannata". Dovreste avere ormai ben chiaro che, a differenza di quanto accade in molti altri programmi per avventure, queste azioni specifiche funzionano esattamente con lo stesso meccanismo di quelle generiche (es. "Prendi"), e cioè con la tavola delle azioni. Anche la suddivisione che ne ho fatto negli appunti è puramente di comodo.
Non vi descriverò in dettaglio tutte le azioni, limitandomi a commentare le più significative e lasciandovi l'utile esercizio di esaminare le altre e capire come funzionano. Comincio subito con le azioni valide in tutti i luoghi dell'astronave, che potete vedere in Figura 1.

L'astronave condannata: azioni valide ovunque

Indossa Casco	2050	-2	goto Prendi XXX
Indossa Tuta	2051	-2	goto Prendi XXX
Indossa Camice	2052	-2	goto Prendi XXX
Lascia Casco	0950		
Lascia Tuta	0951		
Togli Casco	2150		
Leva Casco	6550		

Fig. 1 — L'astronave condannata: azioni valide ovunque (segue).

```
Togli Tuta          2151
Leva Tuta           6551     -10  if luogo(OG)<>0
                                     "Non ce l'hai"
                                  if LU>=9 or (LU=7 & var2=1) {fuori}
                                     "Aaaagh"
                                     Morto
                                  else
                                     goto Lascia XXX

Togli Camice        2152
Leva Camice         6552     -3   goto Lascia XXX

Guarda Tuta         1051     -11  "E' la tua tuta per attiv. xtraveic."

Guarda Secondo      1053     -12  "Ha il casco danneggiato..."

Guarda Cartello     1060
Guarda Manuale      1055     -13  "Non è meglio leggerlo?"

Guarda Camice       1052     -14  "Pesante, trattato al piombo"

Leggi Manuale       2555
Apri Manuale        2255     -15  if luogo(OG)=LU
                                     "Prendilo in mano, prima"
                                  else
                                     "Istruzioni: per attivare..."

Ripara antenna      2769     -29  "Ci vorrebbe il secondo"
```

Fig. 1 — L'astronave condannata: azioni valide ovunque.

Casco, Tuta e Camice

Già le prime tre azioni ("Indossa Casco", "Indossa Tuta" e "Indossa Camice"; tralascio gli articoli, come d'altronde fa il programma) meritano attenzione. Per semplificare il gioco, ho stabilito che prendere uno di questi tre oggetti equivale ad indossarlo (vedi infatti la routine di "Prendi" nel precedente capitolo). Sembrerebbe allora conveniente indicare semplicemente "Indossa" come sinonimo di "Prendi" (stesso codice nel dizionario): in questo modo, le frasi "Prendi Tuta" ed "Indossa Tuta" avrebbero esattamente lo stesso effetto. Disgraziatamente, non sempre

la semplicità paga: con questo sistema, sarebbero accettate frasi come "Indossa Manuale" o "Indossa Chiave". A questo proposito, ci sono due punti di vista: il primo è quello (tipico di molti autori) del "Chi se ne frega? Nessuno dirà mai una frase del genere". Io ritengo invece che il realismo e l'interesse di un'avventura si misurino anche e soprattutto sulla cura dei dettagli. Per fortuna, il parser e la tavola delle azioni rendono semplice l'aggiunta di nuove possibilità, e così anche i pigri possono costruire un'avventura ricca di dettagli senza troppa fatica.
Ad esempio, è molto facile fare in modo che "Indossa Casco" abbia lo stesso effetto di "Prendi Casco". Basta mettere nella tavola delle azioni la sintesi di "Indossa Casco":

Luogo:	00 (dovunque)
Indossa:	20
Casco:	50

La sintesi è dunque 2050, numero da inserire al debito posto nella tavola (che, lo ricordo, va in ordine di numero di sintesi). Di seguito va messo il numero dell'azione (routine Basic) da eseguire. Dato che la frase deve avere lo stesso effetto di "Prendi Casco", basta mettere lo stesso numero della routine di "Prendi", cioè —2 (il segno meno, come sempre, vuol dire che l'oggetto dev'essere presente). Ho commentato con "goto Prendi XXX" per ricordare che questa azione ha lo stesso effetto di quell'altra. Lo stesso discorso vale ovviamente per "Indossa Tuta" e "Indossa Camice".

A questo punto, la frase "Indossa Casco" ha lo stesso effetto di "Prendi Casco", ma la frase "Indossa Chiave" non è riconosciuta da nessuna parte e produce quindi la risposta "Non capisco". In un'avventura ben curata, sarebbe il caso di prevedere anche un default (cioè una risposta standard) per il caso "Indossa XXX", cioè per tutti i tentativi di indossare oggetti non indossabili (ad esempio: "Non sapevo che tu potessi cambiare forma. Puoi anche trasformarti in un verme?"). Meglio ancora: una risposta adatta dovrebbe essere prevista per ogni verbo, quando non sia usato nel modo o sull'oggetto corretto. Altrimenti, la frase "Non capisco" diventa presto noiosa; la varietà mantiene invece l'interesse del giocatore. Sempre a proposito di Casco e Tuta, osservate (in Figura 1) le frasi "Lascia Casco", "Leva Casco", ecc. Dato che è irrealistico consentire che un astronauta si levi il casco o la tuta senza danni mentre si trova nel vuoto, bisogna considerare questa possibilità: se il personaggio compie l'azione all'esterno dell'astronave (luogo>=9), o nel compartimento stagno (luogo 7) quando questo è aperto verso il vuoto (variabile 2, come poi vedremo), accade l'inevitabile. Altrimenti, si esegue la normale

routine di ”Lascia XXX”. Per farlo, basta eseguire un GOTO in Basic alla routine in questione, come si può vedere alla linea 2220 del programma. Le linee 2210-2230 costituiscono la routine di azione.
Una nota interessante: la parola ”Leva” usata in queste frasi funziona come verbo in questo caso, e come nome in altre situazioni (nella sala reattore). Il parser non si pone problemi: gli basta trovare nella tavola delle azioni la sintesi dei codici delle parole, la posizione non gli interessa (si potrebbe fare anche un ”Leva Leva”).

In un luogo e in tutti i luoghi

Anche se in questa avventura non ci sono esempi adeguati, vi ricordo che una stessa azione può avere effetto diverso in un certo luogo piuttosto che nel resto dei luoghi in generale. Ad esempio, la frase ”Chiama Aiuto” potrebbe funzionare soltanto vicino alla caverna degli orsi (a voi immaginare l'effetto), e negli altri posti rispondere con un desolante ”Nessuno è in grado di sentirti”. In questo caso, si può certamente scrivere una routine valida in tutti i luoghi (la cui sintesi comincerà quindi con 00) ed effettuare un confronto (IF LU=47 THEN...) all'inizio della routine per stabilire se ci si trova nel luogo speciale, ma è meglio usare due routine distinte: una generica (valida in tutti i luoghi), che dà una risposta di default, ed una specifica (valida in un solo luogo), che realizza l'effetto desiderato. Ci pensa il parser a dare la precedenza a quest'ultima se l'avventuriero si trova nel luogo indicato. Vi ricordo che l'ordine di precedenza delle varie azioni è descritto nel Capitolo 8.

Azioni specifiche

La Figura 2 mostra l'elenco delle azioni che hanno effetto soltanto se la frase è pronunciata in uno specifico luogo. Se la frase viene pronunciata in un luogo diverso da quello indicato, il parser non la trova, allora guarda se la tavola contiene la stessa azione valida in tutti i luoghi. Se proprio non c'è, stampa ”Non capisco”.
Notate che i numeri delle azioni (routine Basic) da eseguire sono tutti positivi: dato che in nessuna di queste azioni la seconda parola indica un oggetto mobile (potrebbe anche succedere, in un'altra avventura), e dato che il luogo è noto (la sintesi della frase comincia col numero del luogo in cui l'azione deve avere effetto), non c'è alcun bisogno di controllare la presenza dell'oggetto citato (ad esempio l'armadietto). La frase ”Ripara

L'astronave condannata: azioni valide in un singolo luogo

Cabina (1):

Guarda Indicatore	011066	16	"Temperatura.."
Leggi etichetta	012570	17	"SOS Galattico, solo per emergenza"
Premi pulsante	012661	18	"Antenna esterna difettosa"

Corridoio Nord (2):

Leggi Cartello	022560	19	"Scala: ingresso riservato..."
A	020500	20	LU=1 {cabina} if luogo(Secondo)=9 {fuori} "Se solo fosse qui..."
E	020300	21	"Meglio non svegliare i passeggeri"

Corridoio (3):

E	030300	21	"Meglio non svegliare i passeggeri"

Corridoio Sud (4):

Leggi Cartello	042560	22	"O: Attne: st. depressurizzata..."
E	040300	21	"Meglio non svegliare i passeggeri"

Cabina Secondo (5):

Apri armadietto	052267	23	if luogo(Manuale) <>-99 "Già fatto" else if luogo(Chiave)<>0 "Chiuso a chiave" else "Fatto" luogo(Manuale)=LU luogo(Camice)=LU

Cabina Capitano (6):

Apri armadietto   062267   24   "Vuoto"

Compartimento stagno (7):

```
E                  070300   37   if var2=0 {chiuso}
                                    LU=4 {corridoio}
                                 else
                                    goto 1 (Direzioni)

O                  070400   38   if var2=1 {aperto}
                                    LU=11 {esterno}
                                 else
                                    goto 1 (Direzioni)

Premi Pulsante     072661   25   "Premi Rosso o Premi Verde"

Premi Rosso        072664   26   if var2=1 {aperto}
                                    "Click"
                                 else
                                    "Si apre..."
                                    var2=1
                                    if luogo(Tuta)<>0 or luogo(Casco)<>0
                                       "Aaaggh"
                                       Morto
                                    if oggetti mobili in LU
                                       luogo(oggetti)=-99
                                       "Persi nello spazio"

Premi Verde        072662   27   if var2=0 {chiuso}
                                    "Click"
                                 else
                                    "Si chiude..."
                                    var2=0
                                    if luogo(Secondo)=0
                                    and luogo(Chiave)=-99
                                       "Rinviene"
                                       luogo(Chiave)=LU
```

Sala controllo reattore (8):

Leggi Cartello   082560   28   "Emergenza in agguato: portate con voi.."

Guarda Indicatore   081066   16   "Temperatura.."

```
Premi Pulsante     082661   30   "Premi Rosso, Giallo o Verde"

Premi Rosso        082664   31   "Click"
                                 "Tubatura che perde"
                                 if var 1=0
                                    var 1=1
                                 else
                                    var 1=0
                                    timer 1=timer 1 div 2

Premi Giallo       082663   32   "Click"
                                 if var 1=1
                                    var 1=2
                                    if luogo(Camice)<>0
                                       "Ti senti poco bene"
                                 else
                                    var 1=0
                                    timer 1=timer 1 div 2

Premi Verde        082662   33   "Click"
                                 if var 1=2
                                    var 1=3
                                    if luogo(Camice)<>0
                                       "Eccesso di radiazioni"
                                       Morto
                                 else
                                    var 1=0
                                    timer 1=timer 1 div 2

Spingi Leva        082465   34   "Clank"
                                 if var 1=3
                                    Fine! "Complimenti"
                                 else
                                    var 1=0
                                    timer 1=timer 1 div 2

Tira Leva          082365   35   "Clunk"
                                 var 1=0
                                 timer 1=timer 1 div 2
```

Esterno (9):

```
Guarda Antenna     091069   36   "Sembra danneggiata"
```

Fig. 2 — Azioni valide soltanto in uno specifico luogo.

Antenna" è stata messa tra quelle valide ovunque, perché deve rispondere "Qui non c'è" (automaticamente, per effetto del segno meno), facendo così intuire che da qualche parte l'antenna dev'esserci (molti giocatori pronunciano questa frase in cabina di pilotaggio).
La prima frase interessante è "A" ("Alto") nel luogo 2 (corridoio). La sua sintesi è composta nel solito modo, ma indicando un numero di luogo/

Luogo: 02
Prima parola: 05
Seconda par.: 00

La sintesi è dunque 020500, cioè 20500. Dato che un'azione locale ha la precedenza su una generale, quando il giocatore scrive "A" nel luogo 2 non viene eseguita la routine generica di movimento (routine 1), ma quella specifica indicata in questo caso (routine 20). Nella tavola delle azioni, si trova al penultimo posto nella linea 4320. Ricordo ancora che le azioni della tavola sono in ordine di numero di sintesi.
La routine che esegue l'azione (linee 2800-2830) non solo provvede a spostare l'avventuriero in cabina di pilotaggio (LU=1), ma stampa anche un messaggio se il secondo pilota non è stato ancora salvato. Questa è la tecnica normale per far accadere qualcosa durante lo spostamento da un luogo ad un altro.
Una routine simile è la numero 21, che stampa un messaggio ("Meglio non svegliare i passeggeri..."), ma non consente lo spostamento (semplicemente, non cambia LU).
Nell'azione "Apri Armadietto" nel luogo 5 (cabina del secondo), si controlla prima che il manuale non sia già apparso. Solo se è ancora nel "limbo" (luogo —99), si controlla se il personaggio possiede la chiave. Solo in questo caso, appaiono il camice ed il manuale, perché non bisogna dimenticare il primo test, altrimenti se il giocatore ripete la frase "Apri Armadietto" una seconda volta, il manuale ed il camice vengono magicamente trasportati sul posto, dovunque essi fossero. La routine Basic è alle linee 2960-2980, e da adesso smetto di indicarvi le linee. Trovatevele da soli, leggendo i REM.

Il compartimento stagno

Il compartimento stagno è un luogo interessante, anche se non pone problemi ai giocatori appassionati di fantascienza. Dato che può essere in due stati, cioè aperto verso il corridoio o verso lo spazio esterno, occorre una variabile per ricordare in che stato si trova. La variabile, che ho

chiamato var2 negli appunti, è in realtà la variabile Basic V2 (Sì, c'è anche una V1, come avete fatto a capirlo?). Se vale zero, il compartimento stagno è chiuso (nei confronti dello spazio esterno), se vale 1 è aperto. Il pulsante rosso lo apre, il verde lo chiude. Sarebbe assolutamente errato, e non solo inutile, indicare un numero di routine negativo per le azioni "Premi rosso" e "Premi verde"; infatti, il parser cercherebbe gli oggetti con codice in dizionario uguale alle parole "Rosso" e "Verde", che non esistono: esistono solo due oggetti con codice "Pulsante", e precisamente gli oggetti numero 12 e 13 (vedi capitolo 7, Figura 4).
All'apertura del compartimento stagno, il risucchio dell'aria nel vuoto porta con sè tutti gli oggetti eventualmente lasciati per terra. Per bontà degli autori, normalmente questo non capita. Sarebbe stato sadicamente facile congegnare l'avventura in modo da avere per terra qualche oggetto essenziale al momento dell'apertura del compartimento stagno. A proposito, è ovvio che nel vuoto non è possibile sopravvivere senza tuta e casco. Mi rendo conto, nello scrivere queste righe, di aver trascurato una possibilità: se l'avventuriero lascia per terra il secondo pilota prima di richiudere il compartimento stagno, questi non rinviene. Nella risposta alla frase "Premi Verde", bisogna prevedere un "if luogo(Secondo)=0 **or luogo(Secondo)=LU....**" Nessuno dei collaudatori c'è inciampato, ma è meglio prevederlo, altrimenti la risoluzione dell'avventura può diventare impossibile.

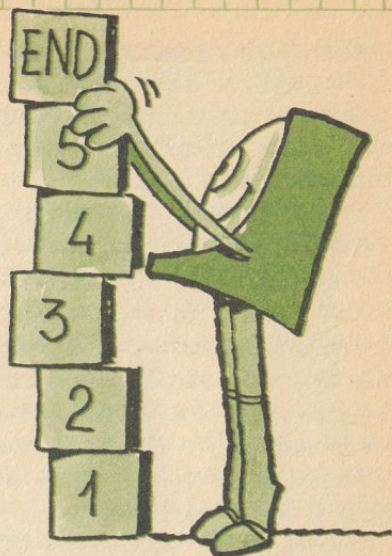
Il reattore

La sala controllo reattore è probabilmente il luogo dell'astronave dove è possibile la maggiore varietà di azioni. Come ben sapete, l'azione finale dell'avventura consiste nel disattivare il reattore sulla base delle sole informazioni ricavate da un manuale incompleto.
Naturalmente, voi avete capito benissimo che per disattivare il reattore occorre eseguire esattamente a rovescio la sequenza di avviamento. Dato che quest'ultima consiste nel tirare la leva, premere il pulsante verde, premere il pulsante giallo, ed infine premere il pulsante rosso, la sequenza opposta è: premere il pulsante rosso, premere il pulsante giallo, premere il pulsante verde e, ovviamente, **spingere** la leva. Pare che gli avventurieri non troppo esperti abbiano difficoltà ad immaginare una tipica leva da quadro di controllo, del tipo di quelle usate per comandare le gru o le scavatrici.
Tornerò sull'argomento della difficoltà a proposito del collaudo e della messa a punto. Per ora, mi interessa spiegare il meccanismo di funzionamento del quadro di comando:

- Se un comando è dato nella sequenza corretta, incrementa un contatore.
- Se un comando è dato fuori della sequenza corretta, azzera il contatore e dimezza il tempo a disposizione (surriscaldamento per errata manovra).

Del fattore tempo parlerò nel prossimo capitolo: per ora vi basta sapere che la variabile T1 (timer 1) viene decrementata automaticamente ad ogni ciclo; quando arriva a zero, il tutto esplode silenziosamente (nel vuoto). Dimezzando T1 si dimezza il tempo disponibile. Il "div" nell'espressione "timer1=timer1" div "2" significa **divisione intera, cioè** senza resto (come si fa in Basic? Guardate la linea 3830).
Tornando al meccanismo di controllo del reattore, la variabile V1 (var1 negli appunti) è il contatore di disattivazione, che all'inizio del gioco vale zero. L'azione "Premi Rosso" (la prima della sequenza) mette V1=1 solo se precedentemente valeva zero, cioè se non era stata eseguita alcun'altra manovra (o se una manovra errata aveva riportato tutto a zero), l'azione "Premi Giallo" (la seconda) mette V1=2 solo se valeva 1, cioè se era stato premuto il rosso e solo i rosso, "Premi Verde" mette V1=3 solo se valeva 2 cioè dopo la sequenza rosso-giallo, e finalmente "Spingi Leva" termina il gioco solo se V1 valeva 3, cioè solo dopo la sequenza corretta rosso-giallo-verde. Ogni manovra fuori sequenza manda alla linea 3830 del programma, che dimezza il timer T1 ed azzera V1, costringendo a ricominciare daccapo la sequenza.
A proposito: il camice serve soltanto per impedire che un avventuriero smaliziato riesca, osservando sull'indicatore di temperatura l'effetto delle varie manovre (non ci avevate pensato, eh?), a disattivare il reattore senza aver prima salvato il secondo pilota: alla penultima manovra ("Premi Verde"), il personaggio muore per eccesso di radiazioni (così impara ad essere troppo furbo).

MONTAGGIO FINALE

Ultimi dettagli

Definita la trama, preparati il dizionario, la mappa, l'elenco degli oggetti, la tavola delle azioni e le routine che le mettono in pratica, siamo quasi pronti per il montaggio finale dell'avventura sotto forma di programma Basic. Mancano soltanto le routine di introduzione, di fine gioco (in bene ed in male), e di passaggio del tempo. Vediamole.

L'introduzione si trova nel programma alle linee 1540-1670. Può naturalmente avere un diverso numero di linea iniziale, ma occorre ricordarsi di modificare la chiamata dall'interprete alla linea 720. Per prima cosa stampa i titoli ed un messaggio introduttivo all'avventura, poi prepara alcune variabili che l'interprete deve conoscere, e precisamente il luogo iniziale e le variabili speciali usate da questa avventura (T1, V1 e V2, cioè il timer, il contatore del reattore e lo stato del compartimento stagno). La linea 1670 fa appunto questo lavoro: LU=6 significa che l'avventuriero inizia il gioco nel luogo 6 (la cabina del comandante), T1=100 vuol dire che l'avventura va risolta entro 100 turni (salvo manovre errate, che accorciano il tempo disponibile). V1=0 e V2=0 sono istruzioni superflue, dato che in Basic tutte le variabili numeriche contengono zero alla partenza del programma, ma non costa molto indicarlo comunque in modo esplicito. Nella routine di "Save", occorre ricordarsi di registrare su disco o cassetta non solo LU e LO%(), ma anche queste variabili T1, V1 e V2. Lo stesso discorso vale per la routine di "Load" (vedi linee 2000 e 2060, la sintassi dipende dal computer usato).

Il finale positivo (avventura risolta) non presenta problemi: basta stampare l'adeguato messaggio di congratulazioni e terminare il programma con l'istruzione END. Questo si può fare direttamente nella routine che

esegue l'azione finale (nel nostro caso, "Spingi Leva"), come si vede alle linee 3640-3770. Il GOSUB 1120 alla linea 3650 serve per evitare che una serie troppo lunga di messaggi scorra fuori dallo schermo prima che il giocatore abba avuto il tempo di leggerla. Usatela pure tutte le volte che lo ritenete opportuno.
Il finale negativo (morto) stampa anch'esso un adeguato messaggio (linee 1200-1320), poi chiede se si vuole ricominciare la partita da capo. In caso affermativo, un semplice "RUN" fa ricominciare il programma, altrimenti un "END" lo termina. Le linee 1330-1360 sono riutilizzabili tali e quali. Notare che la 1340 estrae il primo carattere della risposta, in modo da accettare "S", "SI", "N" e "NO"; altri caratteri iniziali non sono accettati e viene riproposta la domanda.
La routine di "morto" è chiamata soltanto da altre routine del programmatore (es. linea 3130), non dall'interprete. Non ci sono dunque problemi a cambiare i numeri di linea, basta tenerne conto nelle chiamate. Inoltre, dato che la routine termina con "END" o con "RUN", può essere chiamata tranquillamente con un GOTO da qualunque livello di subroutine (non ci sono problemi di RETURN).
Vediamo infine il meccanismo di passaggio del tempo, che trovate alle linee 1400-1500. In caso di modifica dei numero di linea, occorre cambiare anche la chiamata alla linea 780 del parser.
Il timer è stato realizzato su misura per questa avventura, ma il principio è valido per tutti i casi in cui vi sia passaggio del tempo (es. una candela che si consuma). Il parser chiama la routine di tempo ad ogni frase introdotta dal giocatore. La routine decrementa il tempo restante (linea 1400) e poi verifica se è il momento di far avvenire uno degli eventi predisposti, che in questo caso sono alcuni messaggi da stampare (linee 1410-1430) ed il messaggio finale di distruzione quando il tempo è definitivamente scaduto. A parte quest'ultimo caso, la linea 1440 ritorna normalmente al parser.

Montaggio

Ora che ci sono tutti gli elementi, si può procedere alla costruzione del programma completo. Vi suggerisco una procedura per il montaggio di una vostra avventura. È sottinteso che vi conviene fare un "SAVE" ogni tanto, altrimenti mancherà di sicuro la corrente, per la prima legge di Murphy ("Se qualcosa può andare storto, ci va"). Per lo stesso motivo, non salvate sempre con lo stesso nome, ma con nomi e numeri progressivi (es. CAVERNE1, CAVERNE2, ecc.). Se avete la cassetta, re-

gistrate alternativamente sulle due facce. Ogni cinque o sei salvataggi, fate una copia su un altro disco o cassetta. Pronti? Via:

1) Caricate in memoria l'interprete, cioè le linee dalla 100 alla 1160.

2) Introducete le linee con i DATA del **dizionario,** verificando ancora una volta che i vocaboli siano in ordine alfabetico, seguito ciascuno dal proprio codice, e che non manchi "FD" alla fine. Vi conviene usare un numero di linea piuttosto alto, diciamo 8000, per non intralciare poi la scrittura delle azioni.

3) Introducete le linee con i DATA della **mappa,** verificando che siano nell'ordine giusto (a partire dal luogo 1), che ogni descrizione sia seguita dall'elenco dei passaggi e, soprattutto, che quest'ultimo sia corretto (è facile sbagliare con sequenze di 12 cifre). Non dimenticate "FM" alla fine. Usate numeri di linea successivi a quelli del dizionario, lasciando un certo stacco (diciamo 300) per eventuali (cioè sicure) aggiunte successive.

4) Introducete le linee con i DATA delle **azioni** (la tavola delle azioni), controllando che ciascuna comprenda numero di sintesi e numero azione. Verificate molto accuratamente che i numeri di sintesi siano in ordine progressivo, e non dimenticate "FA" alla fine. I numeri di linea devono essere successivi a quelli della mappa, con il solito stacco.

5) Introducete le linee con i DATA degli **oggetti,** verificando che siano nell'ordine voluto (che avete scritto da qualche parte), che siano a gruppi di tre (descrizione, codice nel dizionario, luogo iniziale), e che alla fine non manchi "FO". Vale il consueto discorso per i numeri di linea.

6) Scrivete, di seguito all'interprete (quindi prima dei DATA) le routine di **morto,** di **tempo** e di **introduzione,** lasciando spazio per modifiche. Anche se avete il RENUMBER, vi conviene usarlo il meno possibile, perché vi fa perdere i riferimenti a cui vi siete abituati scrivendo le varie routine. Lo userete alla fine per riordinare il tutto. (Se siete tra i fortunati utenti del Microsoft Basic 2.0 sul Macintosh, non avete il problema dei numeri di linea tra i piedi. Tanto meglio). Sistemate le **chiamate** del parser alle linee 720 e 780, rispettivamente alla routine di introduzione ed a quella di tempo. Se non usate il tempo, scrivete una routine costituita da un semplice RETURN (meglio non alterare il parser). Non dimenticate, nell'introduzione, di mettere in LU il numero del **luogo iniziale.**

7) Scrivete il **commutatore** di azioni, sul modello delle linee 1710-1750. La prima linea (ON A GOTO ...) indica i numeri di linea delle prime die-

ci azioni, la seconda (ON A-10 GOTO ...) quelli delle dieci successive, e così di seguito. La linea 1750 è una **trappola** utile durante la messa a punto del programma: se il numero dell'azione da eseguire è maggiore del numero di linee previste nel commutatore, stampa un messaggio del tipo "AZIONE 25" e ritorna. Per comodità, è utile indicare i numeri di linea delle azioni a passo 50 o giù di lì, in modo da lasciare un certo spazio tra l'uno e l'altro.

8) Scrivete le subroutine che eseguono le **azioni generiche,** che potete copiare dalle mie (azioni 1-7, linee 1770-2120). Dovete modificare "Prendi" (2) e "Lascia" (3), togliendo le aggiunte che riguardano l'astronave. Dovete anche modificare "Save" (5) e "Load" (6) per il vostro computer, se non è un Apple II; non dimenticate di salvare e rileggere anche gli eventuali timer e/o variabili speciali. Naturalmente, potete aggiungere le altre routine generiche da voi previste (es. "Muovi XXX", "Apri XXX", ecc.): più ce ne sono, più l'avventura sembrerà "intelligente" al giocatore. **Importante:** verificate, e se necessario modificate, i numeri di linea indicati nel commutatore. Ciascuno di essi deve corrispondere alla prima linea della routine corrispondente (e non a un REM). Nel mio programma, le azioni 8 e 9 non sono usate, al solo scopo di mantenere raggruppate le azioni generiche, lasciando spazio per eventuali aggiunte. Voi organizzatevi come vi pare.

9) Scrivete le subroutine che eseguono le **azioni specifiche** della vostra avventura, ricordando sempre che devono iniziare al numero di linea indicato nel commutatore alla posizione corrispondente al numero della routine (il primo numero indica dove comincia la prima routine, ecc.).

10) **Verificate** ancora una volta che tutte le chiamate del commutatore ON...GOTO vadano alla prima linea della routine corrispondente, e che ogni routine finisca in ogni caso con RETURN o con GOTO ad un'altra routine. L'avventura è finita: non resta che collaudarla.

Debug

Illusi! Credevate di aver finito, ma non siete che a metà del lavoro (beh, un poco più oltre, ma solo un poco). Rimangono da fare due operazioni che portano via parecchio tempo, ma sono fondamentali per la buona riuscita di un gioco di avventura: il debug ed il collaudo.
Il **debug,** come sapete, è soltanto la ricerca degli errori. Nel caso di un'avventura in Basic, ce ne sono di due possibili tipi: errori di program-

ma ed errori di gioco. I primi sono quelli ben noti, che si manifestano con il famigerato "SINTAX ERROR" (o con altri messaggi più stravaganti) o che causano effetti strani, come può causarne la mancanza di un RETURN in fondo ad una subroutine. A proposito, se dopo aver stampato "Un attimo di pazienza..." il programma si blocca segnalando errore, può darsi che il numero indicato nel messaggio di errore non sia quello effettivo della linea che lo contiene. Infatti, un errore nei DATA può essere segnalato, a seconda dei casi, nella linea effettiva di DATA dove c'è l'errore, in una linea di DATA successiva, o in una linea della routine di lettura (1040-1070). Un motivo in più per controllare bene i DATA. Un altro errore tipico è il riferimento ad un elemento inesistente di un array. Se la vostra avventura prevede più di 100 vocaboli in dizionario, 30 luoghi, 150 azioni o 50 oggetti, dovete modificare le linee 970-1000 che stabiliscono la dimensione massima di ciascun array. Dato che questo non è un corso di Basic, gli altri errori di programma ve li trovate da soli.

Per trovare gli errori di gioco, c'è un solo modo: giocare l'avventura provando, nei limiti del possibile, tutte le combinazioni di eventi e situazioni. La prima cosa da fare è il controllo della mappa: provate tutte le direzioni da ogni luogo, comprese quelle non ammesse, controllando che coincidano con quelle previste nel vostro disegno.

Quando siete sicuri che la mappa è corretta, controllate il dizionario: l'avventura deve riconoscere tutte le parole che avete inserito. Questo serve anche a controllare che abbiate rispettato il corretto ordine alfabetico nei DATA.

Verificate poi che gli oggetti si trovino nei luoghi dove dovrebbero essere, e controllate che siano prendibili soltanto quelli che devono esserlo. Nel fare questo, state anche controllando la routine di "Prendi".

E siamo al passo più importante: il controllo delle azioni. Prendete i vostri appunti e, partendo dalla prima azione, provate tutte le possibili varianti di ogni azione prevista. Non tralasciate alcun particolare ("che, tanto, funziona di sicuro"). Provate proprio tutto. In particolare, collaudate a fondo la "sequenza principale", cioè la catena di azioni che porta alla soluzione dell'avventura. Una famosa software house statunitense ha messo in circolazione un'avventura non risolubile a causa di un errore del genere (ed ha dovuto sostituire parecchi dischetti): capita anche nelle migliori famiglie. Per alleggerire il lavoro di controllo, vi do un aiuto: potete fermare il programma (con CTRL-C, STOP, BREAK o come diavolo si chiama sul vostro computer), cambiare il valore delle variabili (in particolare LU, ad esempio LU=5) e fare CONT. In questo modo potete cambiare lo stato del gioco senza dover fare sequenze complesse di azioni. Questo è specialmente comodo nel caso di avventure di medie

dimensioni, dove occorre molto tempo per procurarsi un certo oggetto o raggiungere un certo posto. Usate estensivamente anche il "Save" per salvare la situazione prima di provare tutte le varianti in un certo luogo (e così collaudate anche "Save" e "Load"). Una nota: se fermate il programma durante la INPUT (linea 800), in alcuni computer non è più possibile ripartire con CONT. Dovete invece scrivere GOTO 800 per ripartire correttamente.
Quando siete sicuri che tutto funziona correttamente, siete pronti per passare alla seconda fase.

Collaudo

Il collaudo è un lavoro che non siete in grado di fare. Non sto sottovalutando la vostra abilità, ma intendo dire che il collaudo di un'avventura (come di qualunque gioco) non può essere fatto dall'autore stesso. Il collaudo serve infatti a stabilire come il gioco viene accolto dai suoi destinatari: i giocatori. Troppi autori tralasciano questo passaggio fondamentale, e mandano in commercio giochi tecnicamente perfetti ma noiosi o ingiocabili (e non solo avventure).
Per il collaudo avete bisogno di uno o più amici, possibilmente non programmatori (o che comunque non hanno seguito il vostro lavoro). Metteteli (uno alla volta) davanti alla tastiera, fate partire il programma e guardate come se la cavano. **Regola fondamentale: non dite nulla!** Non dovete aiutarli a risolvere il gioco, dovete invece prendere appunti e segnarvi tutte le frasi che non avete previsto ma meritano una risposta, le situazioni troppo difficili, quelle troppo facili, quelle "Uffa, che barba", ecc. Scoprirete che quel problema non era facile come sembrava, che quella battuta non era poi così divertente, che la linea di comportamento ovvia era ovvia solo per voi, e tante cose del genere. Scoprirete anche che alcuni risolvono subito il problema A e si piantano sul problema B, mentre altri fanno esattamente il contrario. Se troppi si piantano sul problema C, è il caso di fare qualche aggiustamento.
Soprattutto, ricordate che lo scopo di un'avventura è il divertimento del giocatore, e non il gusto di dire "Nessuno l'ha mai risolta (he, he, he...)". D'altra parte, il superamento di ostacoli troppo facili non dà sddisfazione a chi gioca. L'unico modo per trovare il giusto equilibrio è far giocare più persone possibile, e calibrare accuratamente la difficoltà e gli aiuti.
A titolo di esempio, ecco le principali varianti che il collaudo ha dimostrato necessarie per "L'astronave condannata":

- Accettare anche "Leggi indicatore", non solo "Guarda indicatore".

L'indicatore non ha effetto sulle azioni, ma è fondamentale per la comprensione della trama e per la suspence: deve essere visto (per questo ce ne sono due, tra l'altro).

- Aggiungere un oggetto "Scaletta che scende" nel luogo 4 (fine corridoio S), ovviamente non prendibile. Molti giocatori, infatti, non hanno l'abitudine di esplorare tutto e non trovano subito il reattore (che deve far comprendere il pericolo e costituire un rompicapo).
- L'azione finale "Spingi Leva" è troppo difficile per la media degli avventurieri principianti. Alcuni, poi, immaginano una leva tipo freno a mano e dicono "Abbassa leva" od altre frasi del genere. D'altra parte, non bisogna facilitare troppo la soluzione per non togliere la soddisfazione di averla trovata. Un possibile compromesso può consistere nel prevedere l'azione "Guarda Leva" e la relativa risposta: "È una tipica leva da quadro di comando, che può essere mossa in due direzioni". Se il giocatore non guarda la leva, peggio per lui. C'è scritto nelle istruzioni che "Guarda" è un'operazione fondamentale. A proposito: se volete pubblicare un'avventura, scrivete tre o quattro pagine di istruzioni semplici e chiare, dirette ad un lettore che non ha mai visto un gioco del genere in vita sua.

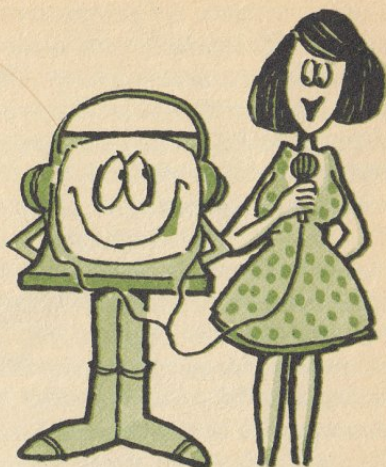
Nel fare aggiunte e modifiche al programma, ricordatevi di rispettare l'ordine alfabetico nel dizionario, e quello numerico nelle sintesi delle azioni. Nuovi luoghi ed oggetti, invece, possono tranquillamente essere aggiunti in coda a quelli già esistenti (non in mezzo, altrimenti cambiano i numeri).

Finalmente

Ora siete in grado di scrivere la vostra avventura. Se qualcosa non vi è chiaro, rileggete con calma il capitolo incriminato, consultate le tabelle riportate in appendice e studiatevi il programma de "L'astronave condannata". Il prossimo capitolo descrive il funzionamento dell'interprete, ed il successivo illustra possibili aggiunte e migliorie per le vostre avventure, ad esempio grafica e suono.

CAPITOLO 12

L'INTERPRETE

Tanto per cominciare

Finora vi ho detto: usate l'interprete senza preoccuparvi di come funziona. Ma spero siate abbastanza curiosi da volerne osservare nei dettagli il meccanismo. È un programma Basic piuttosto breve (circa 150 linee, inclusi i commenti), ma discretamente sofisticato: alcune sue caratteristiche discendono dal programma originale di "Avventura nel castello", altre provengono dal mio studio per il linguaggio ADL-1 (ADventure Language 1). È, o almeno vorrebbe essere, anche un esempio di buona programmazione: routine brevi, ciascuna delle quali svolge un compito ben delimitato. Non è commentato come dovrebbe, ma vediamo subito di rimediare.

Se avete un computer con un Basic decente, scoprirete che molte routine possono essere semplificate. D'altra parte, questo programma è stato scritto per essere **portatile,** cioè per funzionare anche con i Basic più scalcinati (come quello del Commodore 64, un vero fossile vivente). In particolare, la limitazione di 80 caratteri per linea è parecchio fastidiosa.

Descriverò il programma partendo dall'inizio e seguendone il flusso logico, con digressioni di tanto in tanto per descrivere le varie routine. Pronti? RUN!

La linea 140 rimanda alla 710, cioè al programma principale (main program, main per gli amici). Perché il programma principale sta in fondo? Per un motivo pratico: nella maggior parte dei Basic, il tempo necessario per trovare la linea indicata in un'istruzione GOTO o GOSUB è tanto più lungo quanto più la linea è avanti nel programma (la ricerca avviene sequenzialmente, partendo dall'inizio). Se ne deduce

che per aumentare la velocità di esecuzione conviene mettere le routine più usate all'inizio del programma. Il programma principale, che non è chiamato da nessuno, può stare tranquillamente in fondo o quasi. In realtà, le mie routine non sono disposte proprio nell'ordine più conveniente, ma anche la chiarezza vuole la sua parte.
Il main (linea 710) chiama per prima cosa la routine di inizializzazione, o preparazione delle variabili, alla linea 970. Questa comincia col dimensionare gli array usati per contenere il dizionario, la mappa, la tavola delle azioni ed i dati degli oggetti (970-1000). Se una o più dimensioni sono insufficienti per l'avventura che volete scrivere, potete tranquillamente aumentarle (compatibilmente con la memoria disponibile). Era senz'altro possibile leggere dai DATA anche il numero di elementi contenuti in ciascun array (es. READ FD), ma ho preferito non farlo e specificare la dimensione con un numero fisso (es. 100 per il dizionario). Questo perché la maggioranza dei compilatori Basic non accettano variabili nei DIM, ma vogliono un numero (che diavolo è un compilatore Basic? E a che serve? Ne riparliamo nel prossimi capitolo).
Gli array numerici sono tutti di tipo intero (come indica il simbolo % che ne segue il nome) per risparmiare spazio, eccetto CA(). Infatti, ogni elemento di array intero occupa di solito 2 byte di memoria, contro i 4-5 di una normale variabile numerica. CA() non può essere di tipo intero, perché deve poter contenere anche numeri di sei cifre, ed il contenuto di una variabile intera è di solito limitato a 32767. Attenzione: non è detto che ci sia convenienza in termini di velocità nell'uso delle variabili intere. Ad esempio, su Apple II e C-64 le operazioni sulle variabili intere sono in realtà più lente delle altre (il valore viene convertito in formato normale, viene eseguita l'operazione, ed il risultato viene convertito all'indietro).
La linea 1010 stabilisce il nome del file da usare per la registrazione ("Save") della situazione corrente. La 1020 non ha alcun effetto, ma ricorda semplicemente che le quattro variabili indicate valgono zero alla partenza del programma (utile in caso di rilettura dopo qualche mese).
Le linee 1040-1070 leggono le informazioni dalle linee di DATA e le mettono negli array appena dimensionati. Dato che sono sostanzialmente identiche, ne descrivo una: la 1040. Per prima cosa, viene letto un dato nella variabile alfanumerica A$; se il dato è "FD", il dizionario è finito ed il programma può proseguire. Altrimenti, cioè se il dato è diverso da FD, la variabile "FD" (numero di dati in dizionario) viene incrementata (c'è 1 dato). L'elemento numero FD (numero 1) dell'array DZ$(), cioè il primo vocabolo, non è altro che il dato appena letto (A$), mentre il dato successivo, cioè il suo codice, può essere letto direttamente dai DATA (con una READ) nel corrispondente elemento dell'array DZ(). Avendo così letto un vocabolo ed il suo codice, il GOTO 1040

ripete il ciclo daccapo (termina solo quando incontra la fine del dizionario, indicata dal dato "FD").
Lo stesso sistema è usato nelle linee 1050, 1060 e 1070, con una particolarità nella 1060: avendo letto in A$ un dato alfanumerico (stringa), ed avendo appurato che non è FA, occorre mettere nell'array CA() non una stringa ma un numero. Dato che in A$ è stata ormai letta una stringa, questa va ritrasformata in numero con la funzione VAL(). L'inizializzazione è finita; la 1080 torna al programma principale, e ci torno anch'io.
La 720 chiama una subroutine (già vista) scritta dall'autore dell'avventura, che stampa i titoli, stabilisce il luogo iniziale (LU), prepara eventuali timer e variabili speciali, e ritorna. La 730 chiama la comoda subroutine alla 1120 (Premi <spazio> per continuare), che serve ad evitare che le informazioni su video "scrollino" fuori vista prima che il giocatore abbia avuto il tempo di leggerle. La linea 1130, che aspetta la pressione della barra spaziatrice, dipende dalla versione di Basic, come già visto nel secondo capitolo. Così com'è, va bene per Apple II e Commodore.

Il parser

Le linee 760-930 costituiscono il ciclo di gioco, occupato in gran parte dal parser (o analizzatore sintattico), che forma insieme alle sue subroutine la parte più complessa del programma (qui abita la "intelligenza"). La Figura 1 ne mostra il diagramma di flusso semplificato. Vediamo di seguirlo a grandi linee, prima di analizzare i particolari più interessanti. Per prima cosa, si stampa la descrizione del luogo in cui si trova l'avventuriero (760) e degli oggetti eventualmente presenti (770), poi è chiamata la routine di passaggio del tempo (780), già vista nel capitolo 11.
La linea 800 effettua l'input della frase del giocatore, e la 810 chiama l'analizzatore lessicale che ne ricava il codice della prima parola (verbo). Previo controllo (820) che sia stata effettivamente scritta una parola (e non solo articoli, spazi o apostrofi), la 830 si arrabbia se la parola finisce per "re" (probabilmente è un verbo all'infinito, come "Prendere"), la 840 verifica che la parola sia tra quelle in dizionario e la 850 chiama nuovamente l'analizzatore lessicale per conoscere il codice della seconda parola. Anche questa viene controllata (860), ricordando che può essere nulla (codice zero) se la frase è formata da una parola sola.
Se la seconda parola non è nulla, può essere un oggetto: la 870 ne cerca

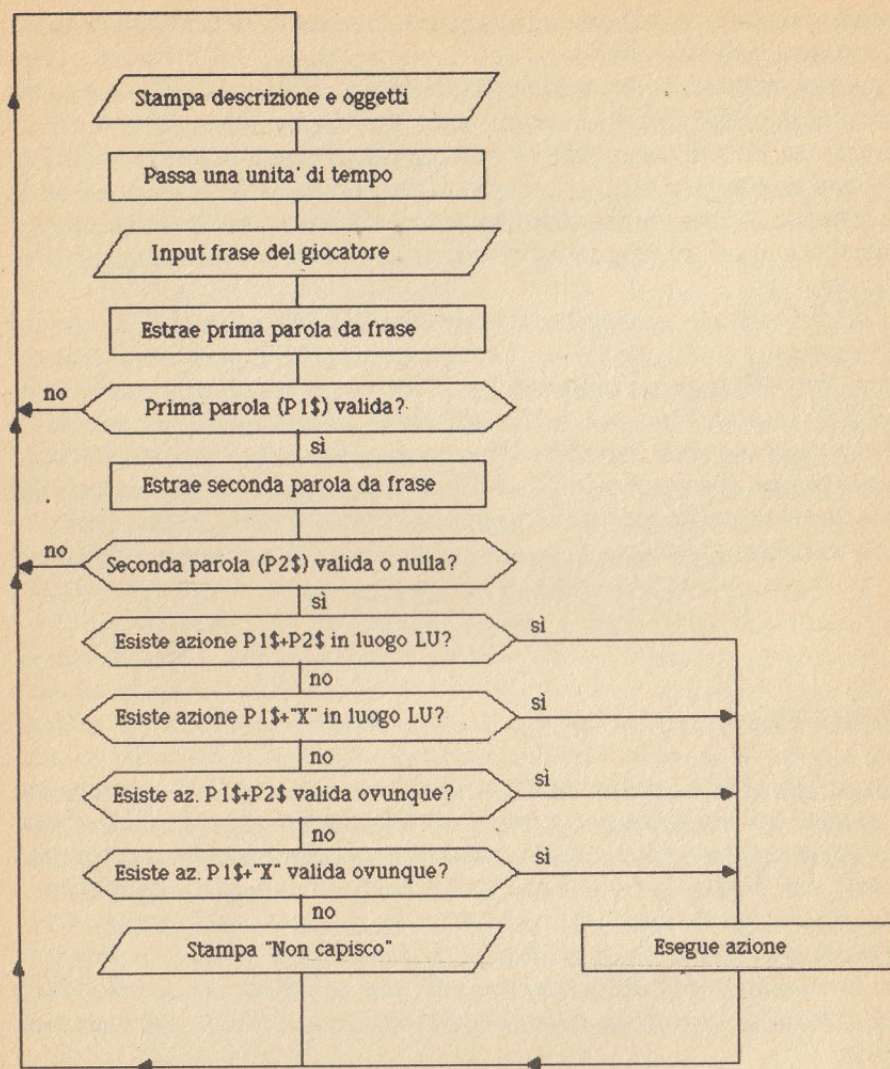



Fig. 1 — Diagramma di flusso semplificato del parser.

l'eventuale indice nell'elenco degli oggetti e lo mette in OG, ad uso sia dell'interprete che del programmatore.
Tutto è pronto: non resta che consultare la tavola delle azioni per vedere se l'azione indicata è stata prevista. La linea 880 prepara i primi due terzi della sintesi di azione, per eseguire le moltiplicazioni una sola volta invece di quattro. Le linee 890-920 cercano nella tavola i quattro possibili tipi

di frase prevista, in ordine di precedenza (leggetevi i REM). Nel caso di frase composta da una sola parola, il codice del nome (C2) è zero e sono eseguite soltanto la 890 e la 910 (le altre non avrebbero senso).
Il test (IF A GOTO 760) al termine di ciascuna linea torna all'inizio del ciclo nel caso in cui la routine 500 abbia riconosciuto un'azione valida e l'abbia di conseguenza eseguita (segnalandolo con A diverso da zero). Se nessuna azione è stata riconosciuta, la 930 stampa "Non capisco" e torna comunque all'inizio del ciclo.

Analisi lessicale

Vediamo un pò di dettagli, cominciando dalla routine alla linea 280, chiamata alla 810 ed alla 840. Questa analizza la frase IN$ introdotta dal giocatore (lunga LI caratteri, vedi linea 810), a partire dal carattere numero IN, ne estrae una parola e consulta il dizionario per stabilirne il codice corrispondente. Notate che il titolo della routine (260) ne riassume il lavoro: la parola viene restituita in P$ ed il codice in C.
La linea 290 salta gli eventuali spazi o apostrofi iniziali, fino ad arrivare all'inizio della parola od alla fine della frase (in questo caso MID$ restituisce una stringa nulla). Se a questo punto il carattere in esame (il carattere numero IN) si trova oltre la fine della frase, non c'era alcuna parola. In tal caso, la 300 provvede a restituire una stringa nulla in P$. Trattandosi di una routine non troppo semplice, ho preferito saltare al RETURN alla 360 invece che metterere un RETURN in mezzo alla routine, che sarebbe senz'altro causa di guai in caso di successive modifiche (dice il saggio: guardatevi dalle subroutine con più punti di uscita e, peggio ancora, da quelle con più punti di ingresso).
La linea 310 segna (in A) il punto d'inizio della parola, le linee 320-330 la scandiscono, un carattere alla volta, fino a trovare un separatore, cioè uno spazio, un'apostrofo, o la fine delle frase. A questo punto, A indica il (punta al) primo carattere della parola, e IN punta al separatore) che termina la parola. La 340 mette la parola in P$ e consulta il dizionario per determinarne il codice, la 350 scarta con disdegno gli articoli (codice 7) e fa ricominciare la ricerca dalla parola successiva, come se l'articolo non esistesse.
La routine di ricerca nel dizionario (180-240) non è altro che la routine di ricerca binaria descritta nel capitolo 20 di **ABC personal computer.** Per i disinformati, il dizionario viene "aperto a metà" e sfogliato scartando la metà che non può contenere la parola (ricordo che sono in ordine alfabetico) e dimezzando ogni volta la parte che può contenerla. In pochi

passaggi, o la parola viene trovata e la linea 190 ne restituisce il codice leggendolo dall'apposito array DZ(), oppure non è in dizionario e la 230 restituisce codice zero. I ed F segnano l'inizio e la fine della zona del dizionario nella quale avviene la ricerca. La linea 180 li pone ai limiti del dizionario, poi la zona viene man mano ristretta. Con 100 parole, il vocabolo viene trovato (o non trovato) entro 7 cicli della routine.

Azioni e oggetti

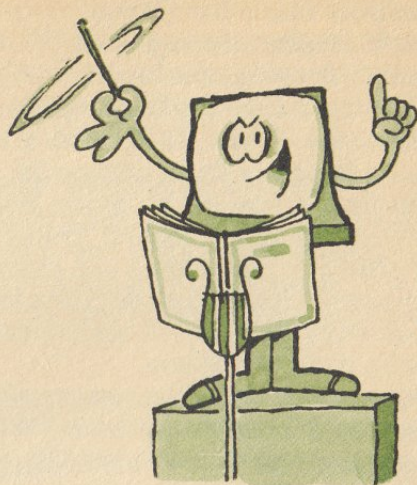
Quando il parser ha preparato una sintesi di frase e vuole sapere se esiste (ed eventualmente eseguire) l'azione corrispondente, chiama la subroutine 500-550 (le chiamate sono alle linee 890-920). Questa routine cerca nella tavola delle azioni, usando la subroutine 400-460 (che fa una ricerca binaria identica a quella del dizionario, ricordate che le azioni sono ordinate per numero di sintesi?). Se l'azione non è stata trovata, la 500 ritorna con A=0, ed il parser passerà ad esaminare la prossima possibilità in ordine di precedenza, o stamperà "Non capisco" se le ha già esaminate tutte. Se invece l'azione esiste nella tavola, A ne contiene il numero. A questo punto, può darsi che occorra controllare l'eventuale presenza dell'oggetto citato (OG): se il codice di azione (A) è positivo, oppure se non c'è una seconda parola, la 510 salta questo controllo. Nel caso contrario (quindi c'è una seconda parola ed il codice di azione è negativo, la 500 mostra che non può essere zero), la 520 controlla che l'oggetto sia presente. Se OG vale zero, non c'è alcun oggetto e la routine ritorna con A=1, cioè come se fosse stata correttamente eseguita un'azione. Infatti, è stata stampata la frase "Qui non c'è", ed il parser non deve fare ulteriori ricerche nella tavola.
La sospirata linea 530, che esegue l'azione, viene raggiunta solo in due casi:

- L'azione è stata trovata nella tavola ed ha codice positivo.
- L'azione è stata trovata nella tavola ed ha codice negativo, e la seconda parola si riferisce ad un oggetto presente.

Nel secondo caso, la linea 520 ha cambiato segno al codice, che è quindi in ogni caso positivo. La 530 esegue l'azione A chiamando il commutatore alla 1710, e poi torna indicando missione compiuta (A=1).
La routine 590-610 stampa l'elenco degli oggetti presenti nel luogo L, ciascuno preceduto dalla frase contenuta in P$. Viene usata dal main (770) per elencare gli oggetti presenti insieme all'avventuriero, preceduti da "Vedo" (es. Vedo un pulsante rosso), e dalla routine di inventario (2120) per elencare gli oggetti presenti nel luogo zero, cioè posseduti

dall'avventuriero. La routine si limita a scandire l'elenco degli oggetti per controllare se il luogo dove ciascuno di essi si trova coincide con L. La funzione ABS() serve per ignorare l'eventuale segno meno davanti al numero di luogo, che significa "non prendibile" (es. il letto).
La breve ma importante routine 650-670, chiamata dal parser alla 870, è quella che mette in OG il numero (indice nell'array) dell'oggetto citato come seconda parola, ma solo se questo è presente o trasportato (altrimenti mette OG=0). Funziona in modo simile alla precedente, scandendo l'elenco degli oggetti per cercare se ne trova uno rispondente alle condizioni richieste: codice in dizionario uguale a C2 e luogo uguale ad LU o a zero. Senza questa routine, sarebbe impossibile riferirsi ad un oggetto nell'ormai nota maniera IF LO%(OG)...

CAPITOLO 13

VARIAZIONI SUL TEMA

Compilare

Quest'ultimo capitolo è dedicato alle possibili variazioni, aggiunte e migliorie che si possono fare ad un programma di avventura. La prima e più semplice operazione consiste nel compilare il programma Basic, cioè nel trasformarlo in un programma equivalente in codice macchina, o comunque in una forma più efficiente. Questo lavoro viene svolto automaticamente da appositi programmi chiamati, per l'appunto, **compilatori Basic.** Sono programmi complessi, che richiedono quasi certamente l'uso di un'unità a dischetto (disk drive). Per ogni computer di grande diffusione esiste almeno un compilatore Basic, e per alcuni c'è un'ampia scelta (per l'Apple II ce ne sono parecchi). Ad esempio, per l'IBM/Olivetti potete usare il BASCOM, per l'Apple II il migliore rimane tutto sommato il vecchio TASC, e per il Commodore 64 c'è il discreto PETSPEED. I vantaggi forniti da un compilatore sono sostanzialmente due:

- La velocità di esecuzione aumenta notevolmente (anche più di 10 volte).
- Il programma originale (o "sorgente") non è più leggibile.

Il primo vantaggio inizia a farsi sentire con avventure di una certa dimensione, e soprattutto con molti oggetti. Il secondo è forse più importante: il programma non può più essere letto e modificato da chiunque, ed il vostro lavoro rimane protetto (non dalla copiatura, ovviamente). Inoltre, tutti i commenti sono rimossi dal programma e non occupano spazio nel compilato (cioè nel risultato finale). In compenso, quasi tutti i compilatori aumentano considerevolmente le dimensioni del programma: se avete problemi di memoria, siete probabilmente nei guai.

Considerando che i compilatori sono profondamente diversi l'uno dall'altro, vi fornisco una regola empirica: per un programma di media grandezza, le dimensioni del codice compilato sono più o meno uguali a quelle del programma originale ampiamente commentato. Partendo da un programma senza REM (cioè scritto da cani), le dimensioni aumentano in genere da un minimo del 20-30% fino ad oltre il doppio dell'originale. Se non volete o non potete compilare il programma, potete almeno usare una utility di "crunch" (compattazione, compressione). Si tratta di un programma che, come minimo, toglie tutti i REM ed unisce quante più linee possibile in una sola linea Basic. Il programma diventa così molto più piccolo, leggermente più veloce in esecuzione e, soprattutto, di difficile lettura.
In ogni caso, compilazione o compressione, commentate ampiamente la versione iniziale del programma. Dato che poi i commenti verranno tolti, non vi costa quasi nulla e semplifica molto il lavoro, specie in sede di modifica.

Migliorie all'interprete

Il programma può essere migliorato in molti modi. Ad esempio, per consentire di salvare (su disco o su nastro) più situazioni con nomi diversi. Basta modificare adeguatamente le routine di "Save" e di "Load". Inoltre, sarebbe il caso di tener conto dei possibili errori durante salvataggio e rilettura (es. disco pieno, disco protetto dalla scrittura, disco leccato dal gatto, ecc.), evitando fastidiose inchiodature del programma.
Una nota per chi usa Apple II con Prodos: non usate il comando STORE per salvare le variabili. Gli array stringa non letti dai DATA introducono periodiche, antipatiche, pause nel programma (garbage collection). Lasciate le cose come stanno.
Se vi sentite esperti programmatori, provate ad aggiungere un controllo del numero massimo di oggetti trasportabili. Sembra facile, ma non basta contare gli oggetti presi con "Prendi" e quelli lasciati con "Lascia". Bisogna anche tener conto di tutti i cambiamenti di luogo, sparizioni, ecc. Naturalmente, non vorrete fare questo controllo ogni volta che un oggetto viene spostato in una routine di azione (es. LO%(7)=13). Vedete un po' di inventare qualcosa di meglio.
Alcuni giocatori preferiscono scrivere in prima persona (es. "Premo il bottone"). Io trovo più simpatica la seconda persona, anche perché introduce quel tanto di schizofrenia (dissociazione mentale) e conseguenti conflitti psichici tra "la mente" (il giocatore) e "il braccio" (il personaggio)

che contribuisce al divertimento. Se proprio volete la prima persona, non avete che da cambiare i verbi nel dizionario. Attenzione ai riflessivi ("Mi Butto"): occorre ignorare il "Mi". Potete anche tagliare la testa al toro ignorando l'ultima lettera del verbo, ma attenti agli irregolari (es. "Fai", "Faccio").

Infine, una possibilità interessante: l'effetto di alcune azioni può dipendere anche dal caso (grazie alla funzione RND()). Come sempre, si tratta di un'arma a doppio taglio: se la cosa non è ben studiata, si ottengono comportamenti del tutto idioti, come quelli dei personaggi in "The Hobbit" (il gioco, non il libro!). Se usata con parsimonia, può invece contribuire a vivacizzare il gioco.

Colore e suono

Se il vostro computer può scrivere a colori sul video, avete l'occasione di aggiungere un poco di varietà e vivacità ai messaggi stampati dal programma. Potete usare un colore per le descrizioni, uno per gli oggetti, uno per l'input ed uno per i messaggi, riservandovi improvvisi cambiamenti di colore o di sfondo per situazioni speciali (ad esempio, sfondo nero nello spazio). È facile modificare il programma a questo scopo, basta inserire le istruzioni di cambiamento del colore di carattere (che sono regolarmente diverse tra un computer e l'altro) nei punti appropriati:

- Per le descrizioni, all'inizio della linea 760.
- Per gli oggetti, all'inizio della linea 590.
- Per l'input, all'inizio della linea 800.

Visto il vantaggio di svolgere un determinato lavoro in un solo punto del programma? Le modifiche sono molto più semplici. Per i messaggi, potete cambiare il colore dopo la 800 o rimetterlo sempre a posto dopo averlo modificato nei tre casi precedenti. Potete anche distinguere i messaggi del parser (tipo "Non capisco", "Non ce l'hai", ecc.) da quelli di gioco, cambiando ulteriormente il colore all'inizio del commutatore di azioni (1710), ma penso che il risultato non sia dei migliori, dal punto di vista del gioco.
Oltre al colore, potete mettere qualche effetto sonoro nei punti più significativi dell'avventura, ad esempio il sibilo dell'aria nel compartimento stagno o la sirena di allarme del reattore. Non riempite l'avventura di suoni e rumori, fanno molto più effetto se arrivano inaspettati.

Grafica

Dal punto di vista del programma, un'avventura grafica è identica ad una di tipo testo, come quella descritta in questo libro. L'unica differenza è che le descrizioni dei luoghi sono sostituite da disegni, come pure lo sono (nelle avventure ben fatte) le descrizioni degli oggetti. In compenso, i messaggi di risposta sono in genere molto brevi, dovendo stare nella parte di schermo non occupata dalle figure.

Se ci sono in circolazione molte avventure di tipo testo scritte male, di quelle grafiche mal realizzate c'è una vera inflazione, specialmente sugli home computer più diffusi. Molti credono che qualche disegno (spesso approssimativo) basti a coprire una sostanziale carenza di intelligenza del programma e di fantasia dell'autore. Particolarmente fastidiose sono le avventure in cui i problemi vanno risolti con una sola frase, scritta con le due sole parole ammesse (nessun sinonimo funziona). Ne ho già parlato, ma l'impressione è che molti autori usino la grafica come semplice copertura per lavori scadenti di questo genere. Morale: se non ve la sentite di scrivere una **buona** avventura grafica, rinunciate ai disegni e fate invece in modo che i giocatori si divertano. Se invece ve la sentite, vi regalo qualche spunto su cui lavorare:

- Per prima cosa, considerate che in quasi tutti i computer l'uso della grafica toglie memoria al Basic. Tenetene conto in modo adeguato.
- Un modo poco impegnativo, ma discretamente efficace, per inserire della grafica nelle vostre avventure consiste nel far apparire, di tanto in tanto, immagini a pieno schermo delle scene più significative (ad esempio, la sala reattore dell'astronave). Si tratta semplicemente di caricare l'immagine da disco quando è richiesta; se non avete il disk drive, o questo è lento come la fame, lasciate perdere (se siete esperti, potete lavorare in linguaggio macchina e tenere qualche immagine nei "buchi nascosti" della memoria). La quantità di immagini può essere aumentata se avete a disposizione un programma di utility che vi permette di tenere le immagini in formato compresso (cioè usando meno memoria).
- Un metodo più sofisticato, usato in gran parte delle migliori avventure consiste nell'utilizzare un "linguaggio grafico", nel quale la figura è descritta come insieme di punti, linee, aree, e così via. Ad esempio, un rettangolo viene memorizzato semplicemente con le coordinate di due vertici opposti. Un simile linguaggio, che consente di tenere molte immagini in poca memoria, può essere già disponibile nel computer, in forma più (Macintosh) o meno (IBM, MSX) evoluta, può essere acquistabile su dischetto (es. Graphics Magician per Apple II) o può

essere scritto da voi stessi, se avete la pazienza e la conoscenza per farlo. Si richiede, come minimo, che il computer abbia le istruzioni per disegnare punti e tirare linee. Se non le ha, occorre procurarsi un'adatta estensione di Basic, come il Simon's Basic per il C-64 (lento, impreciso e scritto male, ma almeno si può usare la grafica senza complicati e lentissimi POKE).

- Nelle avventure grafiche, il problema principale è costituito dagli oggetti: è antipatico disegnare un luogo senza far vedere gli oggetti presenti. O affrontate l'ardua strada di sovrapporre gli oggetti al disegno senza pasticci o assurdità prospettiche, o ripiegate su un disegno degli oggetti presenti fuori della figura principale. Per l'inventario, la figura grande può essere sostituita da tante figurine rappresentanti gli oggetti attualmente posseduti.

Qui mi fermo. Ho appena scalfito la superficie del problema, e mi ci vorrebbe un altro libro per trattarlo in maniera adeguata (potrei anche scriverlo, prima o poi...). In sostanza, la strada delle avventure grafiche può dare grandi soddisfazioni e grandi delusioni. Se la volete percorrere, fate prima un pò di sana pratica con classiche avventure di tipo testo. Ricordate che la grafica non può comunque salvare un'avventura scadente.

Conclusione

Qui finisce l'avventura (del signor Bonaventura, per gli antichi lettori del Corrierino). Ho cercato di passarvi una parte della mia esperienza, ora tocca a voi. Da parte mia, ho ancora molto da raccontare e spero di avere presto l'occasione per farlo. Ciao a tutti e buon divertimento!

Enrico Colombini

TAVOLA 1: VARIABILI PRINCIPALI DELL'INTERPRETE

Dizionario:

FD	Numero vocaboli in dizionario.
DZ$(1..FD)	Vocaboli, in ordine alfabetico.
DZ$(1..FD)	Codici dei vocaboli corrispondenti in DZ$( ).

Mappa:

FM	Numero luoghi in mappa.
DE$(1..FM)	Descrizioni luoghi, in ordine di numero.
DI$(1..FM)	Collegamenti del luogo descritto nel corrispondente elemento di DE$( ) con i sei luoghi adiacenti (N/S/E/O/A/B).

Azioni:

FA	Numero azioni nella tavola.
CA(1..FA)	Codici (sintesi) di azione, in ordine numerico.
AZ%(1..FA)	Numero azione corrispondente da eseguire, negativo se è richiesta la presenza dell'oggetto citato.

Oggetti:

FO	Numero oggetti.
OG$(1..FO)	Descrizioni oggetti, in ordine di numero oggetto
CO%(1..FO)	Codice nel dizionario dell'oggetto descritto nel corrispondente elemento di OG$( ).
LO%(1..FO)	Luogo dell'oggetto corrispondente:

1..FM	luogo indicato, prendibile.
—1..—FM	luogo indicato, non prendibile.
0	trasportato dall'avventuriero.
—99	limbo (fuori gioco).

Varie:

LU	Luogo corrente dell'avventuriero (variabile).
OG	Numero oggetto citato come seconda parola, zero se non presente o trasportato.
P1$,P2$	Prima e seconda parola valida della frase.
C1,C2	Codici nel dizionario di P1$ e P2$

TAVOLA 2: SINTESI AZIONE ED ORDINE DI RICERCA

Sintesi di azione = numero di 6 cifre (LLPPSS):

Codice luogo (2) + Codice prima parola (2) + Codice seconda parola (2)

es. 034709:
l'azione viene eseguita se la frase composta dalla due parole con codici di dizionario 47 e 9 è pronunciata nel luogo 3.

Prime due cifre (codice luogo):

00	Azione valida ovunque.
01..FM	Azione valida solo nel luogo indicato.

Cifre centrali (codice prima parola):

01..FD	Codice nel dizionario della prima parola.

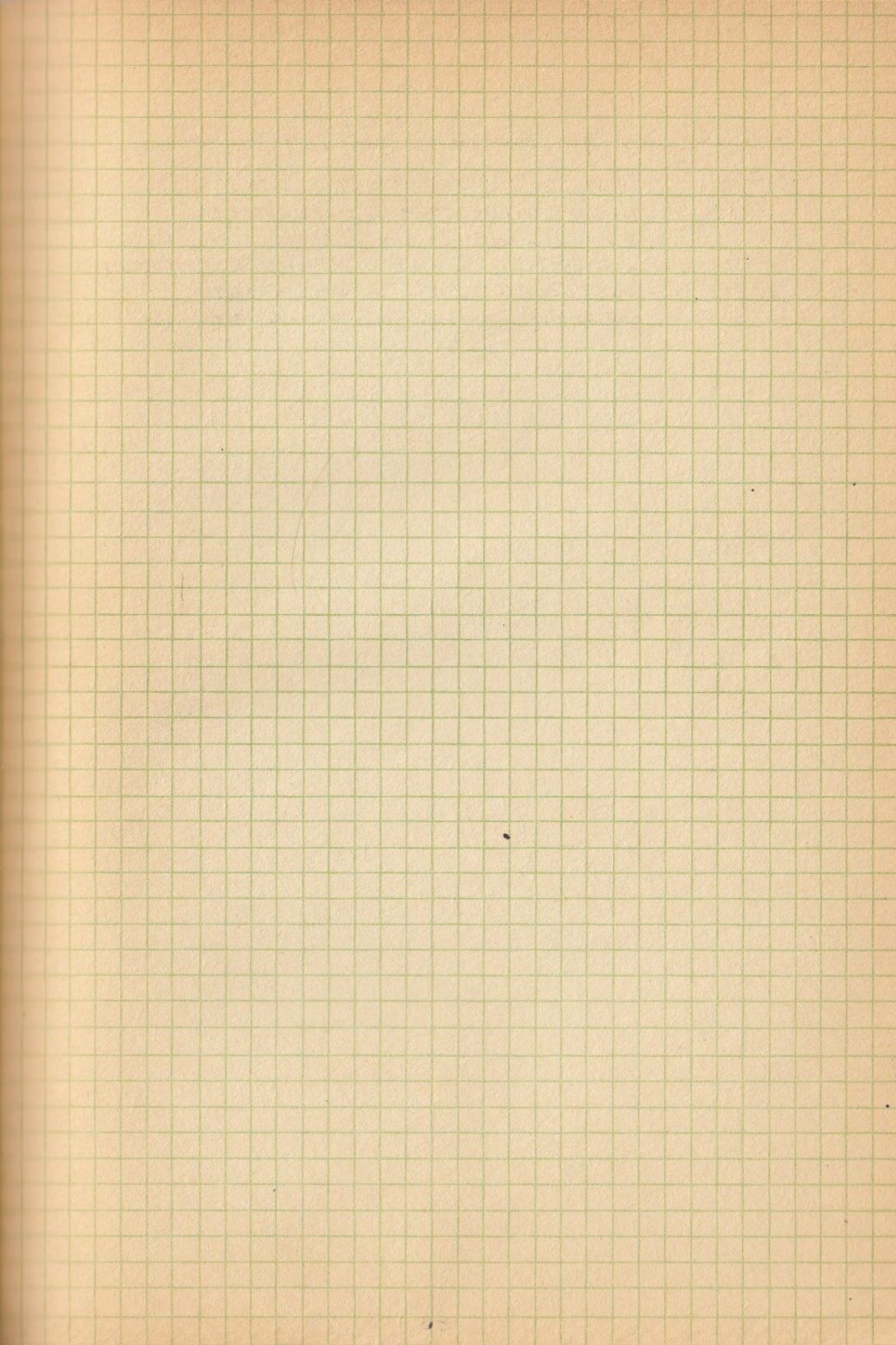
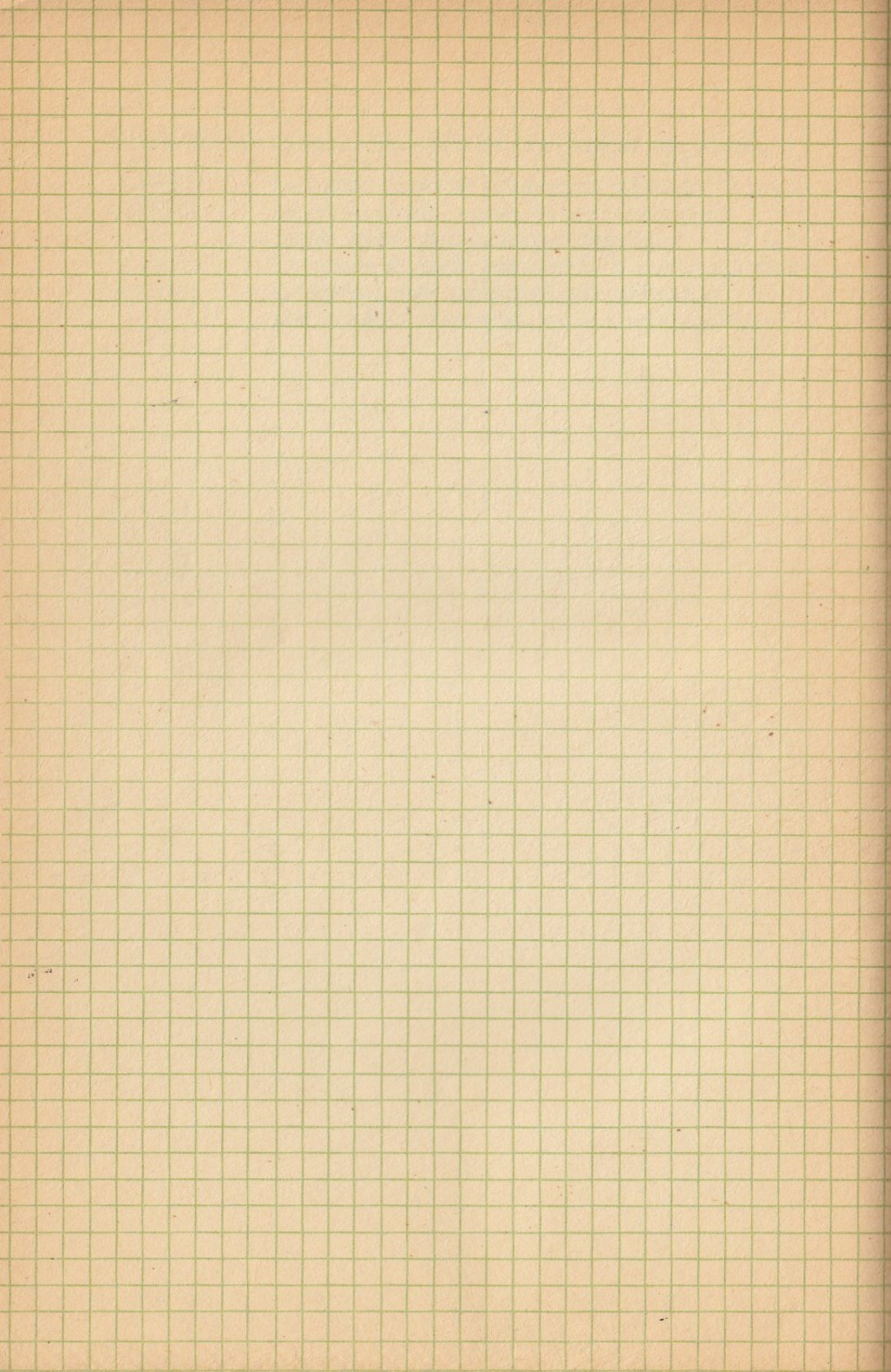
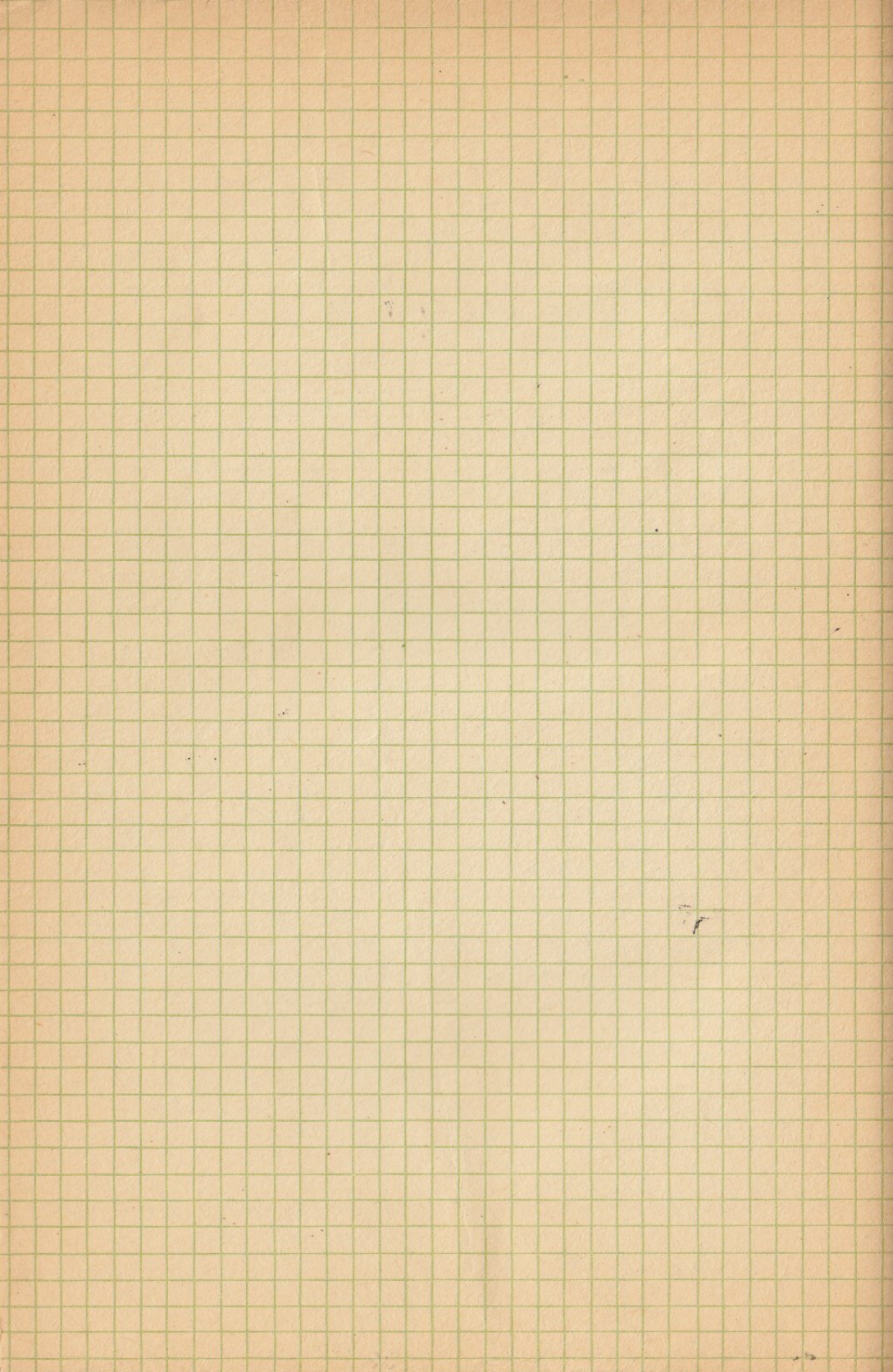
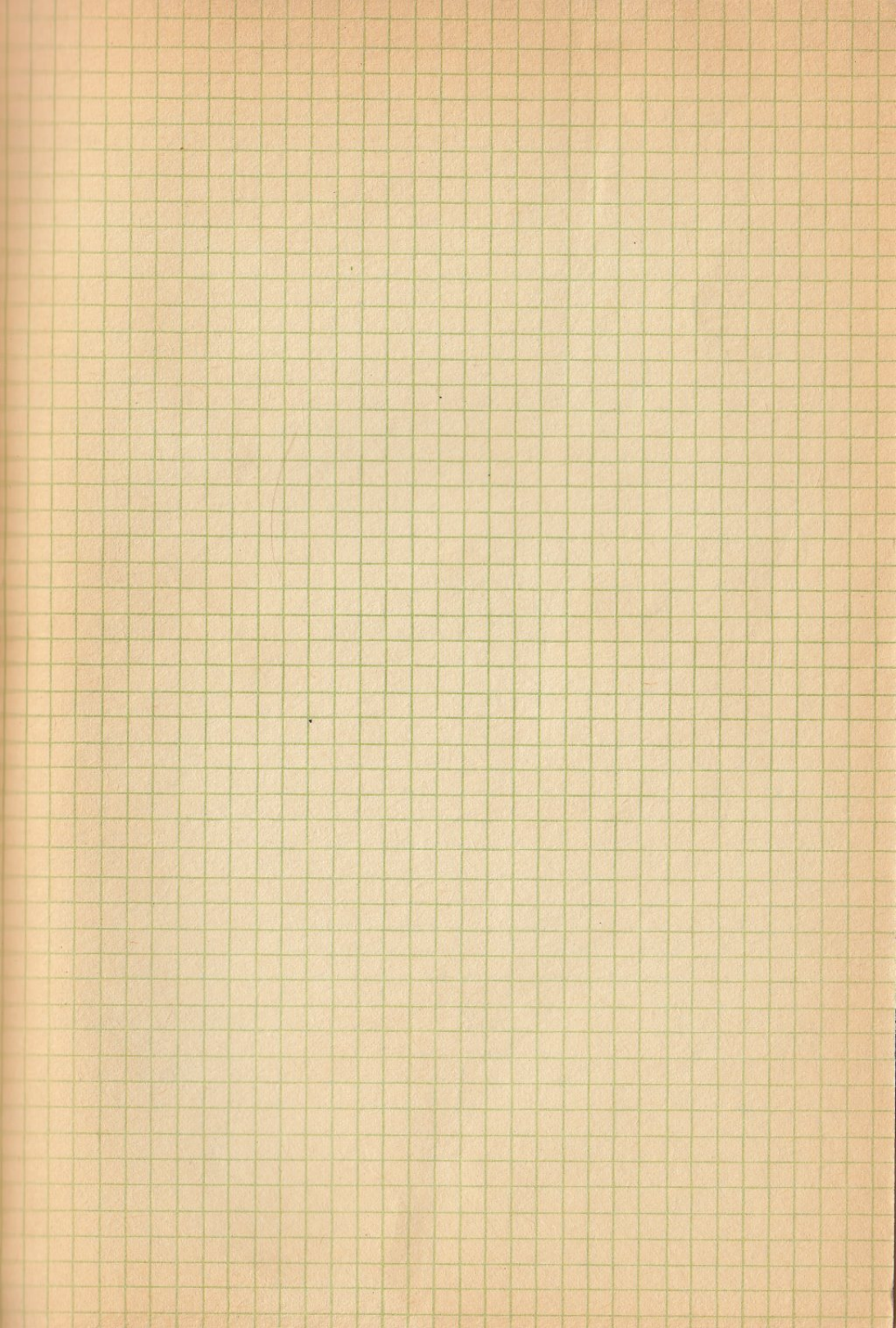
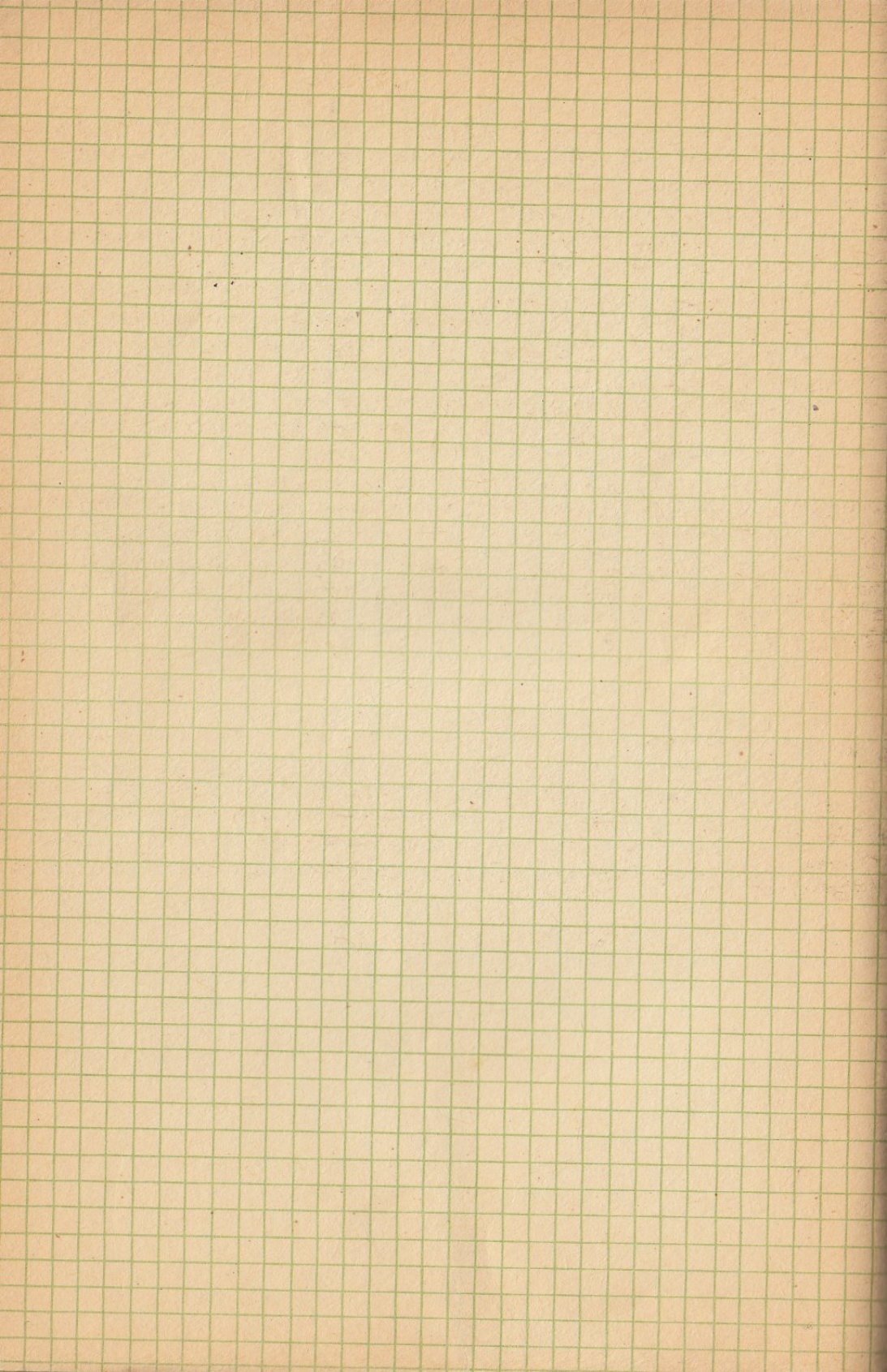
Ultime due cifre (codice seconda parola):

00	Seconda parola nulla, frase di una sola parola.
01..FD	Codice nel dizionario della seconda parola.
99	X: qualunque parola va bene, purché valida, cioè presente nel dizionario e non ignorata (es. articoli).

Ordine di ricerca azioni (precedenza) con frasi di una parola:

LLPP00	La parola pronunciata nel luogo LL.
00PP00	La parola pronunciata in luogo qualunque.

Ordine di ricerca azioni (precedenza) con frasi di due parole:

LLPPSS	La coppia di parole pronunciata nel luogo LL.
LLPP99	La prima parola, seguita da una qualunque parola valida, pronunciata nel luogo LL.
00PPSS	La coppia di parole pronunciata in un luogo qualunque;
00PP99	La prima parola, seguita da una qualunque parola valida, pronunciata in un luogo qualunque.

Questo libro ha tre facce:

- Un'avventura completa da giocare per puro divertimento.
- Una tecnica per costruire giochi di avventura con poca fatica.
- Una lezione di Basic di buon livello.

Dopo aver giocato "L'astronave condannata" (per chi non vuole trasciversi il listato, il programma è incluso in JACKSON SOFT" AVVENTURE per i principali home e personal computer), il lettore viene guidato dietro le quinte del gioco, dove apprende i trucchi del mestiere di un riconosciuto maestro del genere avventuroso. Una minima conoscenza di Basic è sufficiente per costruire le proprie avventure, sfruttando un programma universale predisposto dall'autore. Per i più esperti, una descrizione completa del funzionamento del programma apre la strada alla costruzione di giochi più complessi e sofisticati.

Cod. 066D ISBN 88-7056-311-1